Het koelsysteem van Bavaria

Opleiding
Wiskunde voor de Industrie
Eindhoven

STUDENT REPORT 95-07

Het koelsysteem van Bavaria

C.F.J. den Doelder
F.A. Hanneman

Mei 1995

Het koelsysteem van Bavaria

Ir C.F.J. den Doelder en Drs F.A. Hanneman

Begeleiders: Dr J. Molenaar en Dr R.R. van Hassel
Technische Universiteit Eindhoven

Dit project is uitgevoerd binnen de post-doctorale opleiding 'Wiskunde voor de Industrie'
januari - mei 1995

Inhoudsopgave

1 Inleiding

1.1 Probleemomschrijving

Tijdens het eerste kwartaal van 1995 is in opdracht van bierbrouwerij Bavaria B.V. een project uitgevoerd dat de modellering van het koelsysteem omvatte. Dit project is ontstaan naar aanleiding van problemen die de laatste jaren zijn voorgevallen met betrekking tot de koelinstallatie. Deze koelinstallatie dient met name de warmte af te voeren die vrijkomt bij de verschillende vergistingsprocessen.
In het algemeen voldoet het koelsysteem aan de gestelde eisen maar af en toe schiet het tekort. Voor Bavaria is het moeilijk goede maatregelen te nemen om het koelsysteem te verbeteren omdat binnen het bedrijf weinig bekend is over het koelsysteem. Slechts enkele grootheden worden gemeten. Daar komt nog bij dat het systeem complex is en stukje bij beetje is opgebouwd en uitgebreid.
Gegeven deze situatie is een project uitgevoerd dat als doelstelling had meer inzicht te verschaffen in de werking van het koelsysteem, met name de verschillende ammoniakstromen. Het probleem is aangepakt met behulp van wiskundige modelvorming en heeft geresulteerd in een computerprogramma dat de werking van het koelsysteem simuleert. Met dit programma is het mogelijk om bepaalde karakteristieke grootheden door te rekenen in de tijd. Op grond van de resultaten kunnen vervolgens maatregelen genomen worden om dreigende problemen te vermijden.

1.2 Indeling van het verslag

Het verslag bestaat uit twee delen: in het eerste deel wordt de wiskundige modellering van het koelsysteem behandeld en worden conclusies en aanbevelingen gegeven die volgen uit de resultaten van de simulatie. In deel II komt het simulatieprogramma uitgebreid aan bod.
Het hier beschreven koelsysteem wordt in paragraaf 2 uitgebreider uitgelegd. De modellering ervan is beschreven in paragraaf 3. Wij hebben het programma dat gebaseerd is op dat model gebruikt om enkele karakteristieke inzichtgevende situaties door te rekenen. In paragraaf 4 zijn de resultaten van de simulaties te vinden. De laatste paragraaf van deel I is gewijd aan conclusies en aanbevelingen.
Deel II begint met een gedetailleerde beschrijving van het computerprogramma in paragraaf 6. In paragraaf 7 is een handleiding voor het gebruik van het programma te vinden.

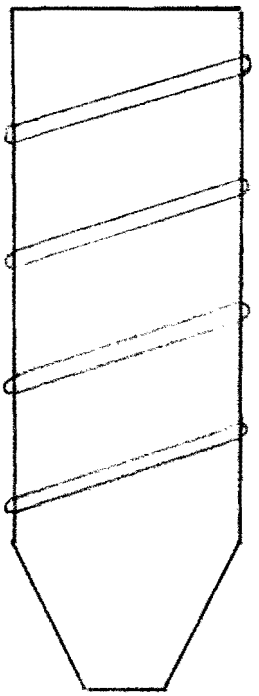
Deel I

Het wiskundige model

2 Beschrijving van het koelsysteem

2.1 Algemene principes

Het hieronder beschreven koelsysteem betreft de koeling van de gist- en lager*apollo's*, grote tanks die het zgn. *wort* bevatten. Wort is een brouwhuisprodukt dat voornamelijk bestaat uit water, graansoorten en hop. Zodra het wort uit het brouwhuis komt, moet het nog twee processen ondergaan voordat het zich bier mag noemen. Allereerst wordt aan het wort gistextract toegevoegd dat wordt vergist. Vervolgens wordt bij een lagere temperatuur nogmaals gistextract toegevoegd en wordt er gelagerd. Lageren is ook een gistproces, onder andere omstandigheden (lagere temperatuur, minder extract). Zowel bij de vergisting als bij de lagering komt warmte vrij. Het koelsysteem moet ervoor zorgen dat het wort niet opwarmt ten gevolge van die warmteproduktie. De apollo's hebben een inhoud van ongeveer 5000 *hl* (zie figuur 2.1). Er zijn ongeveer 100 apollo's, verdeeld over twee groepen. Eén groep wordt voornamelijk gebruikt voor de vergisting, de andere vooral voor de lagering. Het koelsysteem bestaat uit een aantal elementen die hieronder beschreven worden. Het algemene pricipe dat aan de koeling ten grondslag ligt is in feite hetzelfde als dat van een huis-, tuin- en keukenkoelkast: vloeibare ammoniak (NH_3) loopt onder lage druk door spiralen van boven naar beneden langs de apollo's ter koeling van het wort. Het wort geeft z'n overtollige warmte af aan de ammoniak die hierdoor begint te verdampen. Aan het eind van de spiraal zal in het algemeen een mengsel van vloeibare en gasvormige ammoniak overblijven. Het gas wordt aangezogen door *compressoren*, die op het dak van de fabriek staan, en onder hoge druk en temperatuur in de *condensors* weer vloeibaar gemaakt. Vandaar stroomt het naar een *buffervat* dat op zijn beurt de *afscheiders* van vloeibare ammoniak voorziet. Deze afscheiders zijn cruciaal in de ammoniakkringloop. Onverdampte ammoniak uit de koelspiralen rond de apollo's wordt opgevangen in de afscheiders, die zorgen voor de toevoer van vloeibare ammoniak naar de apollo's. Indien nodig worden ze bijgevuld vanuit het buffervat. De inhoud van de afscheiders is van groot belang voor de koelhuishouding. Als de afscheiders te vol raken zuigen de compressoren vloeistof aan, wat ontoelaatbaar is; ze gaan dan kapot. Als ze te leeg raken ontstaat er gevaar voor de apollokoeling, er kan te weinig ammoniak voorhanden zijn om nog aan de vraag te kunnen voldoen. Indien er problemen zijn in de praktijk, blijken de afscheiders heel vaak leeg te zijn.
In figuur 2.2 is het koelsysteem schematisch weergegeven. In de volgende paragraaf worden alle onderdelen van het systeem (met name de cursief weergegeven elementen) afzonderlijk gemodelleerd.



Figuur 2.1: Apollo met koelspiraal

2.2 Processen in de Apollo's

In de apollo's worden twee soorten bier "gemaakt":

- Pils: 3.3 miljoen hl per jaar. Om pils te maken is 14.5 g gistextract per 100 ml wort nodig.
- Zwaar bier: 300.000 hl per jaar. Voor zwaar bier is 20 g gistextract per 100 ml wort nodig.

Het extract is vergistbaar voor 80 %. Bij 1 kg glucose komt 180 $kcal$ (= 752.4 kJ) vrij (het vergistbare deel van het extract is glucose).
Er wordt ongeveer 4 miljoen hl bier per jaar geproduceerd. De minimale productie in een week is 45.000 hl; de maximale productie is 115.000 hl.

De processen die plaatsvinden in de apollo's zijn achtereenvolgens:

vullen
: De tank wordt in ongeveer 5 uur gevuld. Het wort komt binnen bij 10 °C en warmt op tot 11 °C (dit gebeurt in de zgn. gistapollo's); dit duurt 3 à 4 uur. De ruimte-temperatuur (temperatuur van de ruimte waar de apollo's staan) is ongeveer 7 °C.

vergisten

Gedurende vijf dagen vindt de vergisting plaats bij 11 $°C$. Zwaar bier vraagt meer koeling dan pils. Daarbij duurt de vergisting van zwaar bier ook een dag langer. Er vergist namelijk een constante hoeveelheid glucose per dag. De warmteafgifte kan dus constant in de tijd gesteld worden. Bij pils is er na 5 dagen nog 2 $g/100$ ml *extract* over, hetgeen inhoudt dat er 2 $g/100$ ml *glucose* per dag vergist is. Bij zwaar bier is er na 6 dagen nog 1 $g/100$ ml *extract* over, wat neerkomt op een vergisting van 2.5 $g/100$ ml *glucose* per dag. Het overgebleven extract is onvergistbaar. Om het wort op de gewenste temperatuur te houden is een regeling ingebouwd. Als ten gevolge van de vergisting de temperatuur opgelopen is tot 11.2 $°C$, wordt de koeling aangezet. De ammoniak welke door de koelspiralen van de apollo's stroomt, verdampt en onttrekt hiermee warmte aan de omgeving zodat het wort wordt afgekoeld. Als de worttemperatuur onder 10.8 $°C$ komt, wordt de koeling uitgezet. Dit gaat zo door tot de vergistingstijd voorbij is.

reductie

Vervolgens twee dagen reductietijd, waarin bijna geen vergisting meer plaatsvindt. In de reductietijd wordt niet gekoeld.

diepkoelen (gist)

In één dag wordt vervolgens gekoeld van 11 $°C$ naar 7 $°C$. Hierbij staat de koeling continu aan.

overpompen

Het wort wordt in ongeveer 5 uur tijd van de gistapollo naar de zogenaamde lagerapollo ter lagering overgepompt. De inhoud van een lagerapollo is doorgaans groter dan die van een gistapollo, zodat in één lagerapollo de inhoud van meer dan één gistapollo past. De ruimtetemperatuur is nu 1 $°C$.

lageren

Er wordt in de lagerapollo 10 % extract toegevoegd, dus ongeveer 500 hl. Bij de vergisting hiervan komt warmte vrij. Gedurende vijf dagen blijft de lagerapollo op ongeveer 7 $°C$. Dit proces is analoog aan proces *vergisten.*

diepkoelen (lager)

Daarna wordt gekoeld van de heersende temperatuur, ongeveer 7 $°C$, tot -1 $°C$. Dit gebeurt in twee dagen.

verblijf

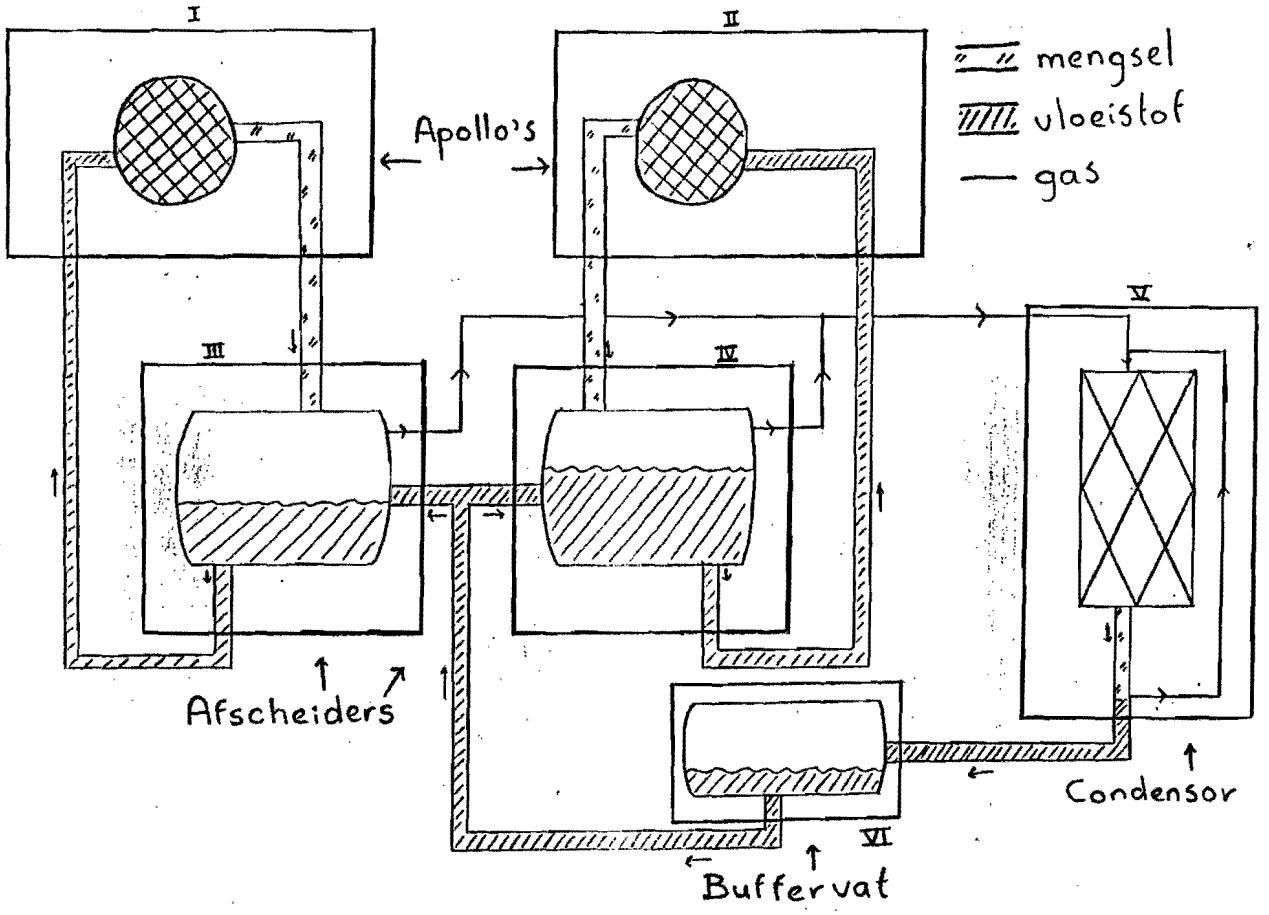
Afhankelijk van de vraag naar bier, blijft de wort nog minimaal vier dagen en maximaal twee weken bij -1 $°C$.

schoonmaken

Als het wort uit de apollo is, wordt deze gereinigd met een water-zeep-oplossing bij

10 $°C$. Wij kunnen er vanuit gaan dat er gedurende ongeveer een uur contact is tussen water en wand. De ammoniak in de koelingsleidingen verdampt nu.

Alleen bij de processen *vergisten, diepkoelen* en *lageren* wordt gebruik gemaakt van de koeling. De lagerapollo's en de gistapollo's staan in aparte delen van de ruimte. Deze delen zijn van elkaar gescheiden door isolerende "gordijnen", zodat het verschil in ruimtetemperatuur tussen beide gehandhaafd kan worden. De ruimte wordt zelf ook gekoeld.



Figuur 2.2: Schema van het koelsysteem

3 Modellering

Het doel van het project is een computerprogramma te schrijven dat het koelsysteem simuleert. Hiertoe is het noodzakelijk het systeem te modelleren. Het model moet eenvoudig maar toch realistisch zijn. Dit betekent dat wij niet alle elementen in het systeem tot in al hun details kunnen modelleren. Het systeem is in een aantal blokken opgedeeld en elk blok wordt afzonderlijk gemodelleerd. In de volgende paragrafen worden deze modellen behandeld. Hierbij moet opgemerkt worden dat het doen van aannamen, altijd noodzakelijk bij modelvorming, hier van groot belang is, gezien het gebrek aan gegevens over het systeem. De processen in de apollo's en het gedrag van de ammoniak in de apollo's zijn goed te modelleren. Veel minder is bekend over de condensors. De afscheiders en het buffervat zijn relatief eenvoudig te modelleren, maar over bepaalde ammoniakverdelingen in die elementen is hoegenaamd niets bekend. Ook is het onduidelijk hoe de afscheiders de ammoniak over de apollo's verdelen. Tenslotte merken wij op dat in het model alleen de produktie van pilsener bier wordt meegenomen. Een correctie voor zwaar bier is eenvoudig mee te nemen. Aan het eind van deze paragraaf is een lijst opgenomen met enkele opmerkingen over het uiteindelijke model.

3.1 Opzet

In onze benadering bestaat het systeem uit 6 blokken. Zie hiervoor figuur 2.2.

Blok I

Dit is een blok waarin één of meerdere apollo's staan. Naar het blok loopt een ammoniakleiding met vloeibaar ammoniak van de afscheider (blok III). Van het blok loopt een ammoniakleiding met een mengsel van vloeibaar en gasvormig ammoniak naar dezelfde afscheider. In dit blok wordt voornamelijk gegist, maar er kunnen wat lagerapollo's tussen staan.

Blok II

Een tweede blok apollo's. Hierin vindt bijna alleen lagering plaats; het is mogelijk dat er enkele gistapollo's tussen staan.

Blok III

Een afscheider die verbonden is met het eerste blok apollo's. Er komen twee leidingen binnen. Eén vanaf het buffervat, één van het blok apollo's. Er lopen twee leidingen vanuit de afscheider; één naar de condensor en de andere naar het blok apollo's.

Blok IV

Een afscheider voor het tweede blok apollo's.

Blok V

De condensor waar de gasleidingen vanuit de beide afscheiders naar toe lopen. Er komt gasvormig ammoniak binnen. Dit doorloopt de condensor. Het vloeibare gedeelte dat eruit komt gaat naar het buffervat. Het gasvormige deel gaat opnieuw de condensor in.

Blok VI

Het buffervat krijgt vloeibare ammoniak binnen van de condensor. Er gaat vloeibare ammoniak uit naar de beide afscheiders.

Voor elk blok kunnen wij debietvergelijkingen[1] opstellen. Wij gebruiken de volgende notatie:
L staat voor een vloeistofdebiet
G staat voor een gasdebiet
$\dot{M}$ staat voor een vloeistoftoename in een systeemelement en η is een maat voor de verdeling van de buffervloeistof over de afscheiders. Index b en c hebben betrekking op buffer- dan wel condensordebieten. Parameter η heeft een waarde tussen 0 en 1 en is een maat voor de verdeling van ammoniak uit het buffervat over de afscheiders.

Voor blok I:

$$L_{in}{}^{I} = L_{uit}{}^{I} + G_{uit}{}^{I} \tag{3.1}$$

Voor blok II:

$$L_{in}{}^{II} = L_{uit}{}^{II} + G_{uit}{}^{II} \tag{3.2}$$

Voor blok III:

$$L_{in}{}^{I} + \eta L^{b} = \dot{M}^{III} + L_{in}{}^{I} \tag{3.3}$$

Voor blok IV:

$$L_{uit}{}^{II} + (1-\eta)L^{b} = \dot{M}^{IV} + L_{in}{}^{II} \tag{3.4}$$

Voor blok V

$$G_{uit}{}^{I+II} = G^{c} + L^{c} \tag{3.5}$$

Voor blok VI

$$L^{c} = \dot{M}^{b} + L^{b} \tag{3.6}$$

Met randvoorwaarde:

$$\dot{M}^{III} + \dot{M}^{IV} + G^{c} + \dot{M}^{b} = 0 \tag{3.7}$$

Een vertraging Δt wordt ingebouwd in de tijdafhankelijkheid, bijvoorbeeld voor blok I

$$L_{in}{}^{I}(t_i) = L_{uit}{}^{I}(t_{i+1}) + G_{uit}{}^{I}(t_{i+1}) \tag{3.8}$$

Om de waarden voor L^{I} en L^{II} te vinden gebruiken wij het eerder beschreven model voor de apollo's.

[1] Met debiet bedoelen wij massadebiet, dat wil zeggen de massahoeveelheid die per tijdseenheid door een bepaald oppervlak stroomt

3.2 De apollo's: blok I en II

3.2.1 Processtap *vullen*

Uit de tijd van drie à vier uur die het wort nodig heeft om op te warmen van 10 $°C$ naar 11 $°C$ kunnen wij de soortelijke warmte van het wort berekenen. Wij gaan ervan uit dat het gist gedurende drie uur aktief is en dan $7.05 \cdot 10^8$ J produceert. Hiermee komt de soortelijke warmte van wort op $c_{wort} = 1.881 \cdot 10^6$ $J/K \cdot m^3$.

vergisting

Nu wij de soortelijke warmte van wort kennen, kunnen wij berekenen hoe lang het duurt voordat bij vergisting de koeling weer aanslaat (in de koeling-aan-uit-cyclus). De koeling start bij 11.2 ° C en stopt bij 10.8 ° C, dus $\Delta T = 0.4$ ° C. Als de koeling stopt duurt het dus ongeveer 72 minuten voordat de koeling weer aanslaat, hetgeen door Bavaria bevestigd is.

lagering

Bij lagering start de koeling bij 7.1 ° C en stopt bij 6.9 ° C, dus $\Delta T = 0.2$ ° C. Omdat er tijdens het lagering veel minder warmte geproduceerd wordt, duurt het veel langer voordat het wort is opgewarmd. Hieruit vinden we een tijdsduur van 4 uur, hetgeen overeenstemt met de ervaring bij Bavaria.

De gevonden waarden zijn samen met andere gegevens te vinden in appendix A.

3.2.2 Processtap *vergisten*

Wij modelleren de apollo en de ammoniakspiralen als volgt. De apollo stellen wij voor als een lange ééndimensionale cylinder. De smalle ammoniakbuis loopt langs deze cylinder. Wij nemen aan dat alle geproduceerde warmte bij de ammoniak terecht komt en niet elders verdwijnt. Verder veronderstellen wij de warmteafgifte van de wort aan het ammoniak homogeen over de apollo.
De cylinder begint op $x = 0$ en loopt tot $x = H$. Wij nemen aan dat de ammoniak bij een temperatuur T_1 de cylinder binnenstroomt. Deze temperatuur is lager dan de temperatuur T_2 waarbij de verdamping van het ammoniak plaats vindt. In het eerste gedeelte van de buis zal de geproduceerde warmte ten goede komen aan het opwarmen van vloeibare ammoniak. Pas daarna zal de verdamping plaatsvinden. Ofwel:

- $T(0) = T_1$
- $T(x) = T_2, \qquad x' \leq x \leq H$

x' is het punt waarop de ammoniak begint te verdampen. De warmte-afgifte per seconde in het deel van 0 tot x' is

$$\frac{x'}{H}\dot{Q} \tag{3.9}$$

met $\dot{Q}$ de totale warmte-afgifte per seconde. De warmte-afgifte in het tweede deel is

$$\frac{H-x'}{H}\dot{Q} \tag{3.10}$$

Uitdrukking (3.9) kan geschat worden door

$$\frac{x'}{H}\dot{Q} = c_{NH_3}(T_2 - T_1)\rho v A \tag{3.11}$$

en (3.10) kan geschreven worden als

$$\frac{H-x'}{H}\dot{Q} = \rho v A L(1-\alpha) \tag{3.12}$$

waarbij A de doorsnede van de ammoniakbuis is, c_{NH_3} de soortelijke warmte van ammoniak, ρ de dichtheid van ammoniak, v de ammoniaksnelheid, L de verdampingswarmte van ammoniak en α de fractie vloeistof op $x = H$.
Als wij deze twee vergelijkingen op elkaar delen krijgen wij

$$\frac{x'}{H-x'} = \frac{c_{NH_3}(T_2-T_1)}{L(1-\alpha)} = 5.6 \cdot 10^{-2}\frac{(T_2-T_1)}{(1-\alpha)} \tag{3.13}$$

Bij een klein temperatuurverschil is x' klein ten opzichte van H.

Wij nemen aan dat het stuk tot x' te verwaarlozen is, oftewel $x' \approx 0$. Wij gaan zoeken naar een uitdrukking voor $\gamma(x)$, de fractie gas als functie van de plaats.
Kies een vaste plaats x en bekijk een tijdstap dt. De snelheid van het ammoniak is $v(x) = \frac{dx}{dt}$. De totale massa die voorbijschiet is $\rho(x)Adx = \rho(x)Av(x)dt$. De totale warmteopname is $\frac{dx}{H} \cdot \dot{Q} = \frac{v(x)dt}{H} \cdot \dot{Q}$. De warmteopname per massaeenheid per tijdseenheid is $\frac{\dot{Q}}{H\rho(x)A}$, dus de fractie vloeistof die gas wordt, is $\frac{d\gamma}{dt} = \left(\frac{\dot{Q}}{H\rho(x)AL}\right)$, met

$$\gamma(x) = \frac{m_{gas}}{m_{liq} + m_{gas}} \tag{3.14}$$

waarbij $0 < \gamma(x) < 1 - \alpha$. Wij krijgen

$$\frac{d\gamma}{dt} = \frac{d\gamma}{dx}\frac{dx}{dt} = \frac{\dot{Q}}{H\rho AL} \tag{3.15}$$

Zodat

$$\frac{d\gamma}{dx} = \frac{\dot{Q}}{Hv(x)LA\rho(x)} \tag{3.16}$$

Als wij aannemen dat $v(x)\rho(x)$ constant is in tijd en plaats, kunnen wij deze differentiaalvergelijking oplossen.

$$\gamma(x) = \frac{\dot{Q}}{Hv(x)LA\rho(x)}x \tag{3.17}$$

Wij noemen $v(x)\rho(x)A$ het (massa-) debiet D.
Als wij op $x = H$ kijken en $\alpha = 0.1$ nemen, dan krijgen wij voor het debiet bij vergisten $D = 5.28 \cdot 10^{-2}\ kg/s$ (bij pils) en $D = 6.59 \cdot 10^{-2}\ kg/s$ (bij zwaar bier), hetgeen een realistische uitkomst is. Vergelijking (3.17) geldt voor het geval dat de temperatuur van het wort constant blijft. Dat betekent dat de koelcapaciteit van het ammoniak net genoeg is om de geproduceerde warmte af te voeren maar niet in staat is om de wort ook nog te koelen. In werkelijkheid neemt deze temperatuur af. De koeling slaat aan bij 11.2 $°C$ en stopt bij 10.8 $°C$. De verdamping van de ammoniak wordt dus mede bepaald door het dalen van de temperatuur van het wort. De voorgaande vergelijking is hiervoor makkelijk aan te passen.

$$\gamma(x) = \frac{\dot{Q} + Q/\Delta t}{HLD} x \tag{3.18}$$

waarbij Q de warmte is die nodig is om het wort 0.4 $°C$ in temperatuur te laten dalen, deze waarde kunnen wij halen uit c_{wort}; Δt is de tijd dat de koeling aanstaat. Wij merken op dat de tijddiscretisatie tot gevolg heeft dat niet precies op de aangegeven temperatuurgrenzen kan worden ingegrepen. Wij nemen aan dat dit geen gevolgen heeft voor de kwalitatieve aspecten van de simulatie.

Wij modelleren nu een twee-stapssysteem waarin volledige verdamping van $x = 0$ tot $x = x_{gas}$ plaatsvindt, en er van $x = x_{gas}$ tot $x = H$ alleen gas in de ammoniakleidingen zit. α is dus nul.
Stel dat op punt x_{gas} alle ammoniak is verdampt. Wij hebben de warmteafgifte van het wort homogeen verondersteld, dus het wort wil nog $\frac{H-x_{gas}}{H}\dot{Q}$ energie kwijt. Wij gaan ervan uit dat het wort nog wel al zijn warmte kwijt kan, maar dat de koeling van het wort afneemt. Vergelijking (3.18) voor het geval dat precies op $x = H$ alle ammoniak verdampt is, luidt:

$$D_{opt} \cdot L = \dot{Q} + Q/\Delta t_{opt} \tag{3.19}$$

Het linkerlid van de vergelijking stelt de warmteopname van de ammoniak voor. Neem nu de situatie waar op $x = x_{gas}$ alle ammoniak al verdampt is. Wij krijgen

$$D_{nieuw} \cdot L = \frac{x_{gas}}{H} D_{opt} \cdot L = \frac{x_{gas}}{H}(\dot{Q} + Q/\Delta t_{opt}) \tag{3.20}$$

Het linkerlid van deze vergelijking is gelijk aan

$$\dot{Q} + Q/\Delta t_{nieuw} \tag{3.21}$$

Δt_{nieuw} is de tijd die de koeling aanstaat wanneer de verdamping dus vroegtijdig is afgelopen. Wij verwachten dat $\Delta t_{nieuw} > \Delta t_{opt}$.

3.2.3 Processtappen *reductie, overpompen, verblijf* en *schoonmaken*

Deze processtappen modelleren wij puur als perioden waarin niets anders gebeurt dan dat de koeling uitstaat.

3.2.4 Processtap *diepkoelen (gist)*

Er wordt in een dag gekoeld van 11 $°C$ naar 7 $°C$. Wij geven hier de numerieke waarden die ook uit appendix A volgen, voor pils. De totale hoeveelheid warmte die in die tijd afgevoerd moet worden is

$$W = c_{wort} \cdot \Delta T = 2.82 \cdot 10^9 \ J \tag{3.22}$$

Dit komt neer op een benodigd vermogen

$$\dot{Q} = 3.26 \cdot 10^4 \ J/s \tag{3.23}$$

Met behulp van formule (3.18) schrijven wij

$$1 - \alpha = \frac{2.38 \cdot 10^{-2}}{D} \tag{3.24}$$

Voor $\alpha = 1$ is een debiet $D = 2.65 \cdot 10^{-2} kg/s$ nodig. Dit is van dezelfde orde van grootte als de debieten die in het echte koelsysteem zijn ingesteld.

3.2.5 Processtappen *lageren* en *diepkoelen (lager)*

Deze processen zijn analoog aan de processen bij het vergisten. Bij processtap *lageren* stopt de koeling bij 6.9 $°C$ en slaat aan bij 7.1 $°C$. Wij kunnen, met gewijzigde waarden formule (3.18) gebruiken.
Bij processtap *diepkoelen (lager)* wordt in twee dagen van 7 $°C$ naar -1 $°C$ gekoeld.

3.3 De condensors

Zoals gezegd is over de werking van de condensors weinig bekend. Wij kennen de typenummers en de ideale capaciteiten van de condensors. Deze capaciteiten hangen af van bepaalde factoren. In principe bestaat er een verband tussen de buitentemperatuur en de condensorcapaciteit. Dit blijkt met name uit het feit dat vooral in hete zomers problemen met de koeling ontstaan. Helaas is dit verband niet bekend. Het effect van de buitentemperatuur wordt getemperd omdat de condensors met water gekoeld worden. Daardoor is de zgn. natte bol-temperatuur een maat voor de condensorcapaciteit. Dit geldt niet als het vriest.
Uitgaande van deze gegevens modelleren wij de condensors als volgt. Wij beschouwen de condensors als één condensor met een capaciteit die afhangt van één parameter die wij buitentemperatuur noemen. Het gas dat de condensor per tijdstap binnenkomt, wordt vloeibaar gemaakt al naar gelang de capaciteit van de condensor. In formulevorm: als $G > C$

$$V = C \qquad , \qquad G_{ret}(nieuw) = G - V \tag{3.25}$$

hierbij is G de som van de hoeveelheid gas dat bij de apollo's vrijkomt en de hoeveelheid niet-gecondenseerd gas uit de vorige tijdstap ($G_{ret}(oud)$), C is de condensorcapaciteit en V de hoeveelheid vloeibaar gemaakte ammoniak. Als $G < C$ geldt

$$V = G \quad , \quad G_{ret}(nieuw) = 0 \tag{3.26}$$

3.4 Het buffervat

Al de vloeibaar gemaakte ammoniak die uit de condensors komt, gaat naar het buffervat. In principe blijft het daar totdat de afscheiders een tekort aan ammoniak hebben. In dat geval vult de buffer de afscheiders aan tot hun regelniveau. In de praktijk is de buffer haast altijd leeg of bijna leeg, zodat de afscheiders niet tot hun regelniveau aangevuld kunnen worden. Het is onbekend hoe de buffer zijn ammoniakafgifte dan verdeelt over de afscheiders. In ons model is die verdeling evenredig met het verschil tussen het regelniveau en het momentane niveau van een afscheider en met de grootte van een afscheider. Bovendien blijft de ammoniak die uit de condensors de buffer inkomt minstens één tijdstap in het buffervat.

3.5 De afscheiders

Zoals eerder al opgemerkt is, spelen de afscheiders een centrale rol in de koelcyclus. In het echte systeem staan drie afscheiders, waarvan er twee op een bepaalde manier gekoppeld zijn. Die koppeling hebben wij benaderd door van die twee afscheiders één afscheider te maken. Zo komen wij aan het systeem in figuur 2.2. Deze gekoppelde afscheider bevoorraadt de grootste groep apollo's, die met voornamelijk gisttanks. De losse kleinere afscheider bevoorraadt de groep met vooral lagerapollo's.
Per tijdstap gebeurt er in ons model het volgende in een afscheider: eerst wordt bepaald hoeveel apollo's koeling nodig hebben in de komende tijdstap. Daaruit wordt aan de hand van de ingestelde debieten per apollo de totale ammoniakafgifte van de afscheiders aan de apollo's berekend. Op het moment dat die ammoniak de afscheider verlaat, wordt deze bijgevuld met de vloeibaar gebleven ammoniak die de vorige tijdstap de apollo's was ingestuurd. Mocht de afscheiderinhoud van het vorige tijdstip onder het regelniveau liggen dan wordt bovendien de afscheiderinhoud aangevuld uit de buffer.

3.6 Leidingen

De verbindingen tussen de afzonderlijke blokken worden gevormd door een complex netwerk van leidingen. Wij weten weinig over het gedrag van de ammoniak in de leidingen, met name de verblijf- en doorstroomtijden zijn onbekend. Tussen bepaalde blokken is het effect van de leidingen meegenomen door een vaste tijdvertraging van 10 minuten op te nemen. Bij andere blokverbindingen beschouwen wij de verbinding alsof de blokken direct aaneengesloten zijn.

3.7 Karakteristieken van de modellering en simulatie

- Wij nemen dezelfde debieten als de insteldebieten van Bavaria. In het uiteindelijke programma kunnen de debieten zelf ingesteld worden.
- Wij laten vrij welke verhouding vloeistof/gas uit de apollo's komt. Ook dit is in het programma in te stellen. Wel bestaat er een verband tussen het debiet en α, zoals in paragraaf 3.2 is aangegeven.
- Het effect van de leidingen is op een grove manier behandeld.
- De condensor heeft een vaste capaciteit bij gegeven buitentemperatuur, die hier meer optreedt als parameter dan als fysische grootheid.
- Het buffervat vult zo mogelijk de ammoniak in de afscheiders aan tot hun regelniveau. Als de buffer te leeg is wordt de ammoniak verdeeld evenredig naar afscheidergrootte en -inhoud.
- Wij beschouwen de twee gekoppelde afscheiders als één.
- De totale hoeveelheid ammoniak in het systeem is in te stellen.
- De computersimulatie gaat door bij te weinig (of te veel) ammoniakvoorraad in de afscheiders en stopt pas als een afscheider vol of leeg raakt.

4 Resultaten

Op grond van het gemaakte model is een computerprogramma geschreven dat het mogelijk maakt om bepaalde grootheden gedurende bepaalde tijd uit te rekenen. Dit maakt het mogelijk het gedrag van het koelsysteem te benaderen als functie van bepaalde parameters. In deze paragraaf zullen wij dit met voorbeelden toelichten. Bij de eerste twee simulaties die wij hier bespreken is *init1.dat* gebruikt, bij de derde *init2.dat* (zie figuur 4.1 en paragraaf 7). Het belangrijkste verschil is dat bij de laatste de beginbufferinhoud veel lager is dan bij de eerste .

4.1 Een extreme simulatie

Wij voeren een simulatie uit waarbij in blok I en II 60 % van de apollo's aanstaan; in elk blok doen alle apollo's precies hetzelfde. Dit is geen reële situatie. Deze situatie heeft een zeer duidelijk periodiek gasgedrag met grote amplitude. Hierdoor schommelen de ammoniakniveaus in de afscheiders zeer sterk; zo sterk dat het minimum- en maximumniveau regelmatig overschreden wordt. In een realistische situatie zou hier ingegrepen worden. Wanneer wij nu ook de condensor een stuk slechter maken (slechts 70 % van de capaciteit), lopen de afscheiders leeg en stopt de simulatie. Van elke simulatie die hier wordt besproken geven wij naast de invoerfiles ook de inhoud van afscheider 1 als functie van de tijd (hier in figuur 4.2) weer omdat die een goede graadmeter is voor het gedrag van het koelsysteem.

4.2 Twee reële simulaties

In de eerste reële simulatie bekijken wij een blok I met 10 verschillende en een blok II met 5 verschillende apollo's. De slingeringen in het gaspatroon zijn een stuk kleiner geworden. Dit komt door de hogere mate van planning; de apollo's lopen uit fase. De variatie in de afscheiderniveaus is daarmee ook minder geworden. Wanneer de condensor nu slechter wordt, blijft de simulatie veel langer doorgaan; pas na enkele weken komen de afscheiders in de problemen. De tweede reële simulatie lijkt op de vorige, alleen is er nu nog beter gepland. De apollo's gaan na elkaar aan. Ondanks het feit dat de condensor op 50 % van zijn capaciteit draait en de bufferinhoud lager is, loopt deze simulatie, door de juiste tijdsplanning, toch goed.

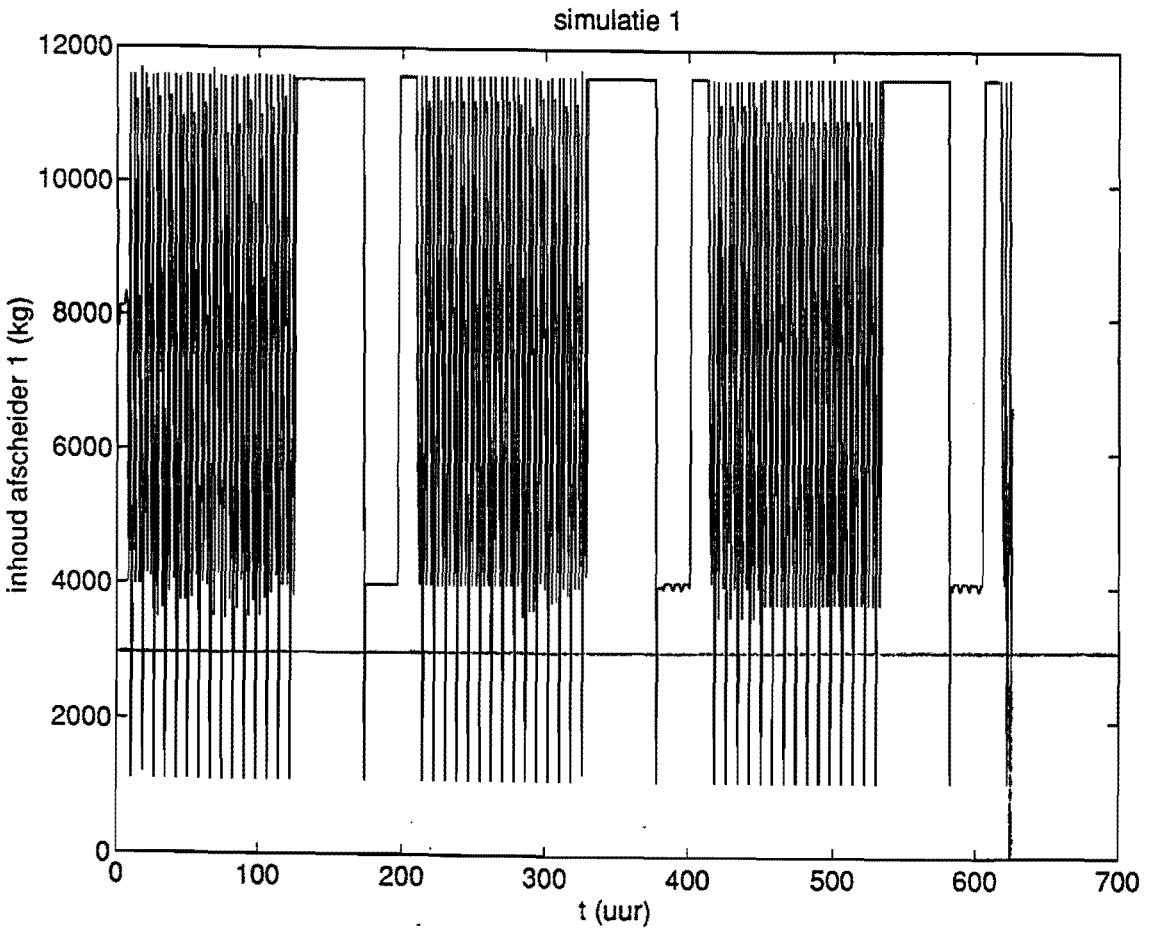
```
Gretour, Tbuiten, Bufinhoud
0 13 7e3
Afschinh[1] en [2]
7813 3906
mininh[1] en [2]
2969 2031
maxinh[1] en [2]
12969 6563
regelinh[1] en [2]
7969 4063
```

```
Gretour, Tbuiten, Bufinhoud
0 13 2e3
Afschinh[1] en [2]
7000 3500
mininh[1] en [2]
2969 2031
maxinh[1] en [2]
12969 6563
regelinh[1] en [2]
7969 4063
```

Figuur 4.1: Init1.dat en init2.dat: definiëren begintoestand van het systeem

```
Gegevens apollo 1:
gist
vullen
180 45.4 10 0.1
Gegevens apollo 2:
gist
vullen
180 45.4 10 0.1
Gegevens apollo 3:
gist
vullen
180 45.4 10 0.1
Gegevens apollo 4:
gist
vullen
180 45.4 10 0.1
Gegevens apollo 5:
gist
niet
180 45.4 10 0.1
```

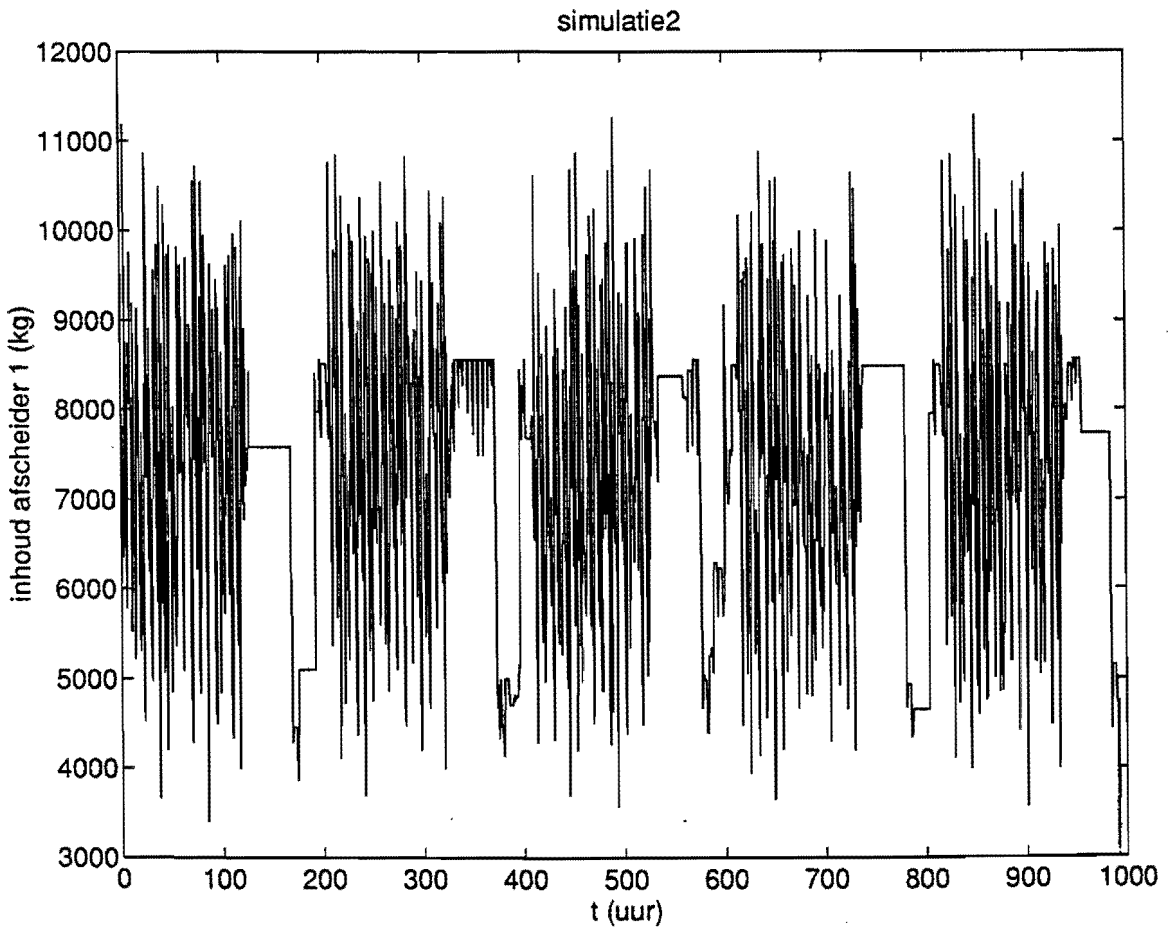
```
Gegevens apollo 1:
lager
vullen
450 16.5 10 0.1
Gegevens apollo 2:
lager
vullen
450 16.5 10 0.1
Gegevens apollo 3:
lager
niet
450 16.5 10 0.1
Gegevens apollo 4:
lager
vullen
450 16.5 10 0.1
Gegevens apollo 5:
lager
vullen
450 16.5 10 0.1
```



Figuur 4.2: Een extreme simulatie; invoerfiles en inhoud afscheider 1

```
Gegevens apollo 1:
gist
koelingaan
180 45.4 10 0.1
Gegevens apollo 2:
gist
niet
180 45.4 10 0
Gegevens apollo 3:
gist
niet
180 45.4 10 0.1
Gegevens apollo 4:
gist
koelingaan
180 50.1 10 0.2
Gegevens apollo 5:
gist
uit
180 65.4 10 0.3
Gegevens apollo 6:
gist
koelingaan
270 52.3 10 0.1
Gegevens apollo 7:
gist
koelingaan
270 60.3 10 0.2
Gegevens apollo 8:
gist
koelingaan
270 50.3 10 0.1
Gegevens apollo 9:
gist
vullen
450 45.3 10 0.1
Gegevens apollo 10:
lager
schoonmaken
450 43.3 10 0.2
```

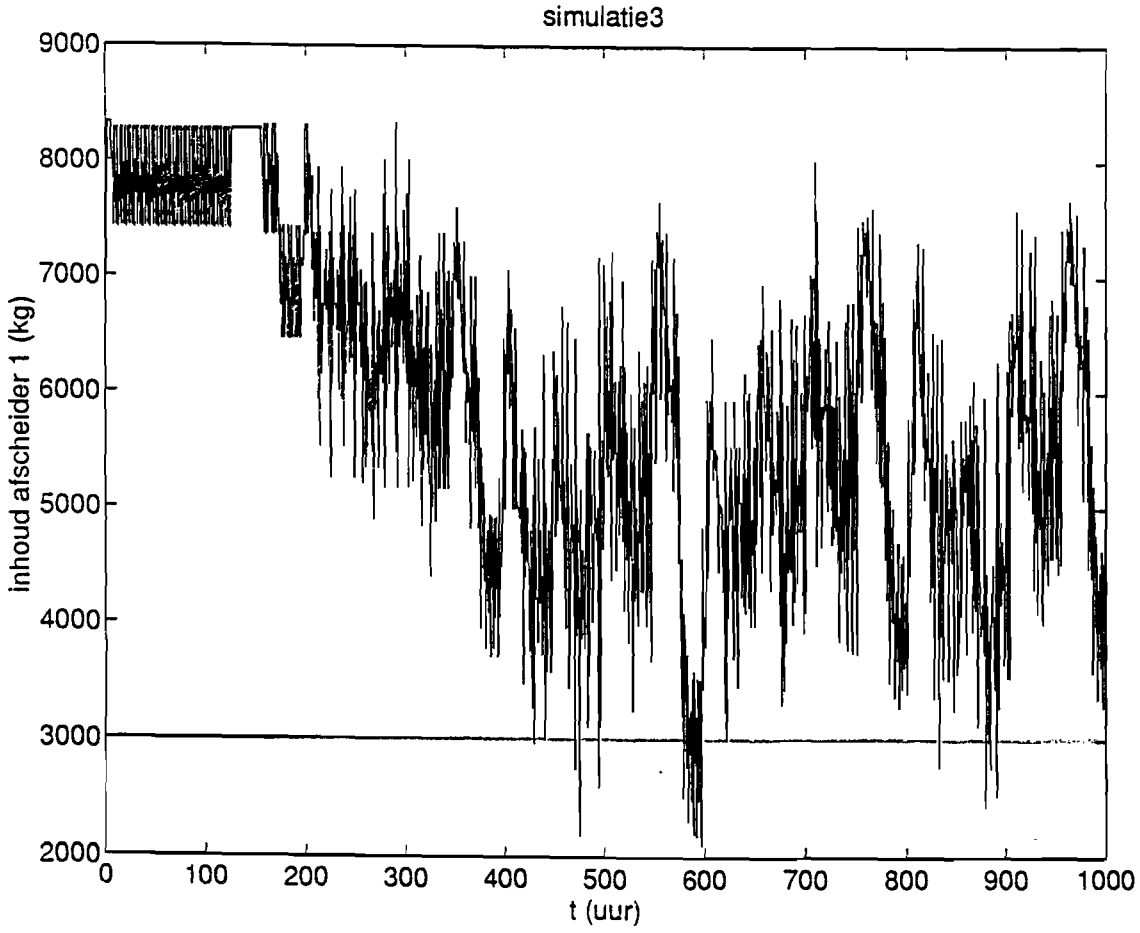
```
Gegevens apollo 1:
lager
koelingaan
450 38.5 10 0.1
Gegevens apollo 2:
lager
diepkoelen
450 50.5 10 0.3
Gegevens apollo 3:
lager
verblijf
450 40.5 10 0.1
Gegevens apollo 4:
lager
koelingaan
450 48.5 10 0.2
Gegevens apollo 5:
lager
uit
450 42.5 10 0.1
```



Figuur 4.3: Een reële simulatie; invoerfiles en inhoud afscheider 1

```
Gegevens apollo 1:
gist
uit
180 45.4 1 0.1
Gegevens apollo 2:
gist
niet
180 45.4 300 0
Gegevens apollo 3:
gist
niet
180 45.4 600 0.1
Gegevens apollo 4:
gist
uit
180 50.1 900 0.2
Gegevens apollo 5:
gist
uit
180 65.4 1200 0.3
Gegevens apollo 6:
gist
uit
270 52.3 1500 0.1
Gegevens apollo 7:
gist
uit
270 60.3 1800 0.2
Gegevens apollo 8:
gist
uit
270 50.3 2100 0.1
Gegevens apollo 9:
gist
uit
450 45.3 2400 0.1
Gegevens apollo 10:
lager
uit
450 43.3 2700 0.2
```

```
Gegevens apollo 1:
lager
uit
450 38.5 3000 0.1
Gegevens apollo 2:
lager
uit
450 50.5 3300 0.3
Gegevens apollo 3:
lager
uit
450 40.5 3600 0.1
Gegevens apollo 4:
lager
uit
450 48.5 3900 0.2
Gegevens apollo 5:
lager
uit
450 42.5 4200 0.1
```



Figuur 4.4: Een goed geplande simulatie; invoerfiles en inhoud afscheider 1

5 Conclusies en aanbevelingen

Het doel van dit project was te komen tot het beter begrijpen van het koelsysteem. Op basis van een wiskundig model hebben wij nu een programma dat het werkelijke systeem simuleert. Door gebrek aan gegevens heeft de simulatie een meer kwalitatief dan kwantitatief karakter. Om dit laatste aspect te verbeteren zijn meer (meet-)gegevens nodig over het systeem. Op grond van de simulaties komen wij tot de volgende conclusies en aanbevelingen.

1 De simulatie geeft een goed beeld van het koelsysteem van Bavaria en kan gebruikt worden als indicator voor het kwalitatieve gedrag van het koelsysteem.

2 Als er een extra vat geplaatst wordt, kan dit het beste als afscheider gebruikt worden.

3 De huidige minimum- en maximumgrenzen van de afscheiders kunnen wellicht scherper worden afgesteld.

4 De ammoniakafgifte van de buffer aan de afscheiders moet op een slimmere manier gekoppeld worden aan de niveaus van de afscheiders.

5 Een adequate planning van de processen in de apollo's heeft een sterk gunstige invloed op de continuiteit van de ammoniakvraag, zodat uitschieters vermeden kunnen worden.

Om verder inzicht in het systeem te krijgen, is het noodzakelijk dat er eerst meer (meet)gegevens beschikbaar komen. Wij komen tot de volgende aanbevelingen die aangeven wat relatief eenvoudig te meten is.

- De reeds bekende meetgegevens zoals de maximum afscheiderinhouden en de bufferinhoud moeten nauwkeuriger gemeten worden.
- Het is nuttig om bijvoorbeeld aan één apollo continu te meten. Zodoende kunnen waarden voor α, het debiet enz. bepaald worden.

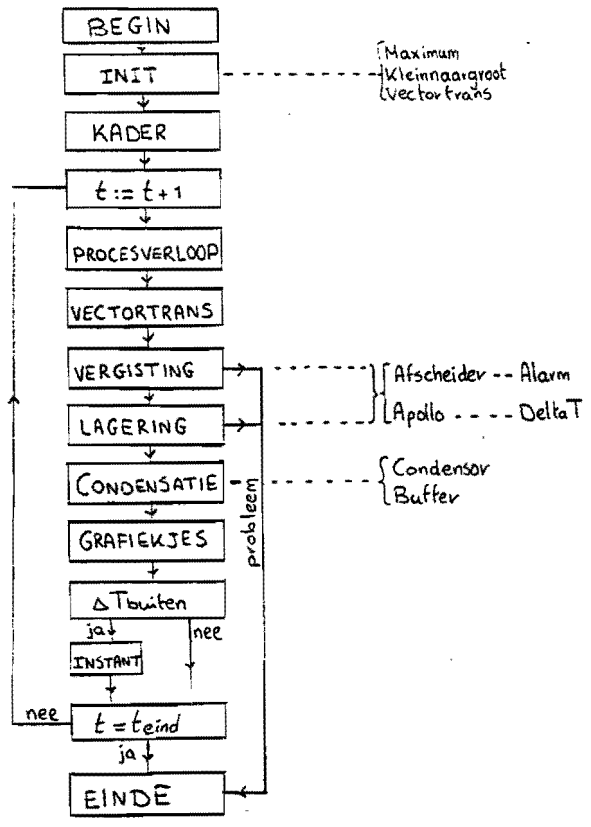
Deel II

Programma

6 Opbouw van het programma

6.1 Algemeen

Het programma *Bavkoel* berekent de ammoniakstromen en -opeenhopingen in een modelkoelsysteem gebaseerd op het koelsysteem van Bavaria. Het programma bestaat uit verschillende procedures en functies die hieronder beschreven worden. Wij hebben gekozen



Figuur 6.1: Schema van de programmaopzet

voor een tijdseenheid van 10 minuten. Dit houdt in dat de systeemgrootheden elke tien minuten opnieuw berekend worden. Observaties van het echte systeem rechtvaardigen deze

tijdseenheid.
Het programma bestaat uit een aantal procedures. Deze procedures berekenen grootheden zoals die in een bepaald gedeelte van het systeem gelden. Voor elke tijdstap worden de procedures doorlopen. Gedurende de eerste tijdstap worden beginwaarden gebruikt die in *Procedure Init* gegeven zijn. Vanaf dan wordt gewerkt met waarden die eerder in het programma berekend zijn.
Allereerst wordt er gekeken welke processen er spelen in de apollo's en of er inmiddels iets veranderd is (*Procedure Procesverloop*). Vervolgens wordt deze informatie gebruikt om te kijken hoeveel apollo's er op dat moment koeling nodig hebben (*Procedure Vectortrans*). Met behulp van deze informatie worden de grootheden in de twee blokken (de gistapollo's en de daarbij behorende afscheider en de lagerapollo's met hun afscheider) berekend. In *Procedure Condensatie* worden de grootheden in de Condensor en Buffer berekend. Een schema van de programmaopzet is te vinden in figuur 6.1.
Wij gaan alle blokken los in het programma verwerken. De blokken I en III en de blokken II en IV worden als één groter blok behandeld. Wij leggen het programma uit aan de hand van een voorbeeldsituatie. Blok I en II laten wij ieder bestaan uit vijf typen apollo's, die verschillen in inhoud of begintoestand; wel wordt het *totale* systeem gesimuleerd door elk type uit meerdere apollo's te laten bestaan. Wij gebruiken codes om een proces aan te geven. Deze staan getabelleerd in tabel 6.1.

code	toestand	koeling	vergisten	lageren
0	Doet niet mee	0	altijd	altijd
1	Uit	0	instelbaar	instelbaar
2	Vullen	0	300 min.	300 min.
3	Koeling uit	0	72 min.	4 uur
4	Koeling aan	1	berekend	berekend
5	Reductie	0	2 dagen	-
6	Diepkoelen	1	1 dag	2 dagen
7	Verblijf	0	-	4 dagen
8	Overpompen	0	5 uur	5 uur
9	Schoonmaken	0	2 uur	2 uur

Tabel 6.1: Codering van het gehele proces

Blok I

In dit blok vindt voornamelijk de vergisting plaats. (Het is mogelijk om in dit blok ook lagerapollo's te plaatsen. Dat doen wij nu niet.)
2 apollo's van 3000 *hl* en 3 apollo's van 5000 *hl*

Processen 3 en 4 nemen samen 5 dagen in beslag.
Bij vergisten krijgen wij het volgende procesverloop:

$$1 \rightarrow 2 \rightarrow 3 \rightleftharpoons 4 \rightarrow 5 \rightarrow 6 \rightarrow 8 \rightarrow 9 \tag{6.1}$$

Als de apollo zich in het begin in toestand 0 bevindt, blijft hij daar in. Toestand 9 gaat over in toestand 2.

Blok II

In dit blok vindt voornamelijk de lagering plaats. (Het is mogelijk om in dit blok ook gistapollo's te plaatsen. Dat doen wij nu niet.)
5 apollo's van 5000 hl

Processen 3 en 4 nemen samen 5 dagen in beslag.
Bij lagering krijgen wij het volgende procesverloop:

$$1 \rightarrow 2 \rightarrow 3 \rightleftharpoons 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \tag{6.2}$$

Toestand 9 gaat over in toestand 2.

Binnen één proces hebben wij een constante α (fractie vloeistof-gas). Dit vereenvoudigt de vergelijkingen voor de blokken I en III aanzienlijk. Sommige processen hebben een vaste tijd (overpompen, schoonmaken, verblijf, vullen, reductie). Bij andere processen hangt de tijd af van de temperatuur van de wort (koeling aan, koeling uit, diepkoelen). Voor de vloeistofdebieten, die in de apollo's stromen, gaan wij uit van de door Bavaria verstrekte gegevens.

6.2 Constanten

Als constanten zijn vaste systeemgrootheden en -eigenschappen (zoals de tijd dat de koeling aanstaat) opgenomen (zie tabel 6.2 en tabel 6.3).

Constanten die betrekking hebben op tijden zijn te vinden in tabel 6.2; de overige constanten staan in tabel 6.3.

6.3 Typen

Procestoestand

Deze heeft als waarde één van de procestoestanden in de apollo's; 'uit', 'vullen', 'koelingaan', 'koelinguit', 'reductie', 'diepkoelen', 'verblijven', 'overpompen', 'schoonmaken', 'niet'.

naam	waarde	omschrijving
koelguit_eind	7	tijd dat koeling uitstaat bij gisten
koelluit_eind	24	tijd dat koeling uit staat bij lageren
vul_eind	30	vultijd voor apollo's
red_eind	288	reductietijd
pomp_eind	30	overpomptijd apollo's
schoon_eind	12	schoonmaaktijd apollo's
hoofd_eind	720	totale gisttijd
verblijf_eind	576	verblijftijd na lageren in lagerapollo's
diepg_eind	144	diepkoeltijd gistapollo's
diepl_eind	288	diepkoeltijd lagerapollo's

Tabel 6.2: Constanten tijdsduren (in eenheden van 10 minuten)

Gistoflager

Gistoflager geeft aan of de betreffende apollo aan het gisten of aan het lageren is; zijn waarde is 'gist' of 'lager'.

Toestand

Dit is een record met daarin verschillende gegevens over een apollo. Het record bevat 'gistlager' (van het type gistoflager), 'proces' (van het type procestoestand), 'koeling' (een boolean die aangeeft of de koeling bij dit proces aan of uit moet staan), 'teller' (houdt de tijd dat het proces tot nu toe duurt bij), 'omslagtijd' (houdt de tijd bij de de koeling aan-uit-cyclus tot nu toe duurt), 'inhoud' (geeft de inhoud van de apollo), 'debiet' (geeft het ingestelde debiet), 'koelaan' (geeft aan of er geswitched moet worden van 'koeling aan' naar 'koeling uit'), 'uittijd' (geeft aan hoe lang de betreffende apollo uitstaat voordat hij mee gaat doen), 'temperatuur' (van het wort) en 'alpha' (de fractie vloeistof-gas uit de apollo).

apollos

Dit is een 2-dimensionaal array, aangegeven met (i,j). Teller i loopt van 1 tot 2 een indiceert om welk blok het gaat. Teller j loopt van 1 tot een bepaald maximum, waarbij elke positie een apollo voorstelt. Een positie in dit array is van het type *toestand.*

punten

Dit type wordt gebruikt voor het grafische gedeelte. Het bevat paren (x, y) voor de vijf grootheden die op het scherm geschreven worden.

naam	waarde	omschrijving
dampwarmte	1.369e6	NH3: warmte opgenomen bij verdampen, eenheid J/kg
gistwarmte	3.918e7	wort: warmte afgestaan bij gisten, eenheid J/10min.
lagerwarmte	7.56e6	wort: warmte afgestaan bij lageren, eenheid J/10min.
soortwort	1.881e6	wort: soortelijke warmte, eenheid J/(K · m3)
nomcap	3900	nominale capaciteit van de condensor, eenheid kg/10min.
maxi	100	maximaal aantal apollo's dat aanstaat
Totafsch1	15625	totale inhoud van afscheider 1 in *kg*
Totafsch2	7969	totale inhoud van afscheider 2 in *kg*
Totbuffer	8e3	totale inhoud van de buffer in *kg*
inh1bav	17888	90 % van de totale inhoud van blok 1 bij Bavaria in *m3*
inh2bav	9945	90 % van de totale inhoud van blok 2 bij Bavaria in *m3*
rho	640	dichtheid van vloeibaar ammoniak in *kg/m3*
maxgas	6000	schaalfactor voor condensorcapaciteit in *kg/10 min*
maxtemp	45	schaalfactor voor buitentemperatuur in graden Celsius

Tabel 6.3: Overige constanten

6.4 Parameters

Wij maken onderscheid tussen parameters die aan het begin ingesteld worden en dan constant blijven en parameters die in het programma zelf nog veranderd worden. De eerste groep staat getabelleerd in tabel 6.4.

naam	omschrijving
blok1aant	aantal apollo's in het vergistingsgedeelte
blok2aant	aantal apollo's in het lageringsgedeelte
maximum	het maximum van *blok1aant* en *blok2aant*
Bufinhoud	volume van de buffer
Afschinh	volume van de afscheiders
Mininh	minimaal toegestane niveau in de afscheider
Maxinh	maximaal toegestane niveau in de afscheider
Regelinh	inhoud waarop de afscheider afgeregeld is
convfactor	conversiefactoren voor de beide blokken

Tabel 6.4: Parameters, die echter binnen de duur van één simulatie constant blijven

De variabelen in tabel 6.5 worden in het programma zelf nog veranderd.

naam	omschrijving
blok	van het type *apollo*
t	tijd
Lin	vloeistofdebiet dat naar resp. blok 1 en blok 2 stroomt
Lin_oud	vloeistofdebiet op vorig tijdstip
Luit	vloeistofdebiet dat uit resp. blok 1 en blok 2 stroomt
Luit_oud	vloeistofdebiet op vorig tijdstip
Guit	gasdebiet dat uit resp. blok 1 en blok 2 stroomt
Guit_oud	gasdebiet op vorig tijdstip
Buit	vloeistofdebiet dat uit de buffer stroomt
Buit_oud	vloeistofdebiet op vorig tijdstip
Regelverschil	verschil tussen regel- en werkelijk niveau in afscheider
Lcond	vloeistofdebiet dat uit de condensor stroomt
Gretour	gasdebiet uit de condensor dat geretourneerd wordt naar de condensor
Tbuiten	buitentemperatuur
uitvoer	uitvoerfile

Tabel 6.5: Parameters die binnen de duur van één simulatie nog veranderd worden

6.5 Beschrijving van de Procedures en de Functies

Wij geven voor elke procedure en functie eerst aan wat zijn invoer is. Wij maken hierbij onderscheid tussen de invoer die binnen deze procedure of functie constant blijft en invoer die veranderd wordt. Vervolgens wordt de uitvoer gegeven en volgt een korte beschrijving van de werking van de procedure of functie

6.5.1 Procedure Afscheider

invoer: (constant)

i (welk blok), *blok[i,j].koeling, blok[i,j].debiet, VAR Afschinh[i], Luit_oud[i], Buit_oud[i]*

invoer: (variabel)

Afschinh[i], Lin

uitvoer:

Afschinh[i]

inhoud:

Deze procedure berekent hoeveel ammoniak er afgegeven moet worden aan de apolloblokken en bepaalt de nieuwe inhoud van de afscheider.

6.5.2 Procedure Alarm

invoer: (constant)
Afschinh[i]

uitvoer:
Melding op het scherm

inhoud:
Deze procedure geeft bij fout melding op het scherm en beëindigt dan het programma. Een fout treedt op wanneer één van de afscheiders helemaal leeg of helemaal vol is.

6.5.3 Procedure Apollo

invoer: (constant)
i (welk blok), *blok[i,j].gistlager, blok[i,j].alpha*

invoer: (variabel)
Luit[i], Guit[i], blok[i,j].koeling, blok[i,j].proces, blok[i,j].temperatuur

uitvoer:
Luit_oud[i], Guit_oud[i], Luit[i], Guit[i], blok[i,j].temperatuur, blok[i,j].koeling]

inhoud:
Deze procedure berekent aan de hand van het inkomende (vloeistof-) debiet en de apollotoestanden de uitgaande gas- en vloeistofdebieten.

6.5.4 Procedure Buffer

invoer: (constant)
Lcond, Regelinh, Afschinh

invoer: (variabel)
Bufinhoud, Buit, Buit_oud

uitvoer:
Bufinhoud, Buit_oud[1,2], Buit[1,2]

inhoud:
Hier wordt de bufferinhoud berekend. Er wordt zoveel ammoniak afgevoerd als nodig is om het afscheiderniveaeau op het regelniveau te houden. Als er niet genoeg ammoniak in de buffer is, wordt de aanwezige hoeveelheid evenredig verdeeld over de afscheiders.

6.5.5 Procedure Condensatie

invoer:

Procedure Condensatie heeft als invoer alle invoer van de procedures Condensor en Buffer.

uitvoer:

De uitvoer van deze procedure is gelijk aan de uitvoer van de aangeroepen procedures

inhoud:

Procedure Condensatie roept eerst procedure Condensor aan en vervolgens procedure Buffer

6.5.6 Procedure Condensor

invoer: (constant)

Tbuiten, Guit_oud[i]

invoer: (variabel)

Gretour, Lcond

uitvoer:

Gretour, Lcond

inhoud:

Deze procedure bepaalt aan de hand van de buitentemperatuur *Tbuiten* de capaciteit van de condensor en berekent dan de hoeveelheid gas die vloeibaar gemaakt wordt. Het gedeelte dat vloeibaar gemaakt is (*Lcond*) wordt afgevoerd naar de buffer.

6.5.7 Function DeltaT

invoer:

blok[i,j].gistlager, blok[i,j].alpha, blok[i,j].debiet, blok[i,j].inhoud

uitvoer:

DeltaT

inhoud:

Deze functie berekent het temperatuurverschil per tijdstap ten gevolge van het ammoniakdebiet door de apollo.

6.5.8 Procedure Grafiekjes

invoer:

ywaarde, kleur, positionering op het scherm

uitvoer:
grafiekjes op scherm

inhoud:
Er worden vijf grootheden naar het scherm geschreven op drie verschillende posities, nl. in de drie kaders.

6.5.9 Procedure Init

invoer:
Drie initialisatiefiles

uitvoer:
Geen

inhoud:
Deze procedure geeft grootheden hun beginwaarden.
Variabelen op nul te stellen: *t, Lin[i], Luit[i], Lin_oud[i], Luit_oud[i], Guit[i], Guit_oud[i], Buit_oud[i],Buit[i], Regelverschil[i], blok[i,j].teller, blok[i,j].omslagtijd, blok[i,j].inhoud, blok[i,j].uittijd.*
Variabelen hun goede waarde geven: *Tbuiten, Bufinhoud, Afschinh[i], mininh[i], maxinh[i], regelinh[i], Lcond, Gretour.*
Apollo's instellen

6.5.10 Procedure Instant

invoer:
Als er een toets is ingedrukt, wordt deze ingelezen

uitvoer:
buitentemperatuur

inhoud:
Deze procedure houdt bij of er tijdens de simulatie een toets is ingedrukt. Wanneer er een 'h' is ingetoetst, wordt de buitentemperatuur 10 graden verhoogd. Wanneer de ingetoetste letter een 'l' is, wordt de buitentemperatuur 10 graden verlaagd.

6.5.11 Procedure Kader

invoer:
kleur

uitvoer:
drie kaders op het scherm

inhoud:

Er worden drie kaders op het scherm getekend

6.5.12 Procedure Kleinnaargroot

invoer:

blok[i,j].inhoud

uitvoer:

conversiefactor voor blok I en II

inhoud:

In de invoerfiles van de apollo's kunnen minder apollo's beschreven staan dan er in werkelijkheid zijn. Om toch een realistische simulatie te doen, wordt er voor beide blokken een conversiefactor berekend.

6.5.13 Procedure Lagering

invoer:

De invoer is gelijk aan de invoer van de aan te roepen procedures

uitvoer:

De uitvoer is gelijk aan de uitvoer van de aangeroepen procedures.

inhoud:

Procedure Lagering roept eerst procedure Afscheider en vervolgens procedure Apollo aan.

6.5.14 Procedure Maximum

invoer: (constant)

blok1aant, *blok2aant*: het aantal apollo's in blok 1 en blok 2

invoer: (variabel)

blok

uitvoer:

Maximum

inhoud:

Deze functie berekent het maximum van *blok1aant* en *blok2aant*. In de rest van het programma lopen alle lussen van 1 tot dit maximum. Voor het blok met de minste apollo's worden de ongebruikte apollo's op 'niet' gezet, d.w.z. dat die gewoon niet meedoen.

6.5.15 Procedure Procesverloop

invoer: (variabel)
blok[i,j].proces

uitvoer: *blok[i,j].proces*
voor het nieuwe tijdstip

inhoud:
Deze procedure berekent de toestand *blok[i,j].proces* van de apollo's op elk tijdstip.

6.5.16 Procedure Vectortrans

invoer: (variabel)
blok[i,j].proces

uitvoer:
blok[i,j].koeling

inhoud:
Berekent welke apollo's koeling nodig hebben aan de hand van *blok[i,j].proces.* Uit *blok[i,j].proces* wordt *blok[i,j].koeling* bepaald.

6.5.17 Procedure Vergisting

invoer:
Procedure Vergisting heeft als invoer alle invoer van de procedures die aangeroepen worden.

uitvoer:
De uitvoer is dezelfde als die van de aangeroepen procedures.

inhoud:
Deze procedure roept achtereenvolgens procedure Afscheider en procedure Apollo aan

7 Handleiding voor het programma

Het programma beschrijft de ammoniakstromen in een model-koelsysteem. Er worden vijf grootheden afgedrukt op het scherm; de afscheiderinhouden, de bufferinhoud, "Guit"(de hoeveelheid gas die uit de apollo's komt) en de buitentemperatuur. Het model-koelsysteem bestaat uit evenveel apollo's als het werkelijke systeem, met het eerste kan echter veel meer gevarieerd worden. De gebruiker kan aangeven voor wat voor soort koelsysteem het programma moet gaan rekenen. De gegevens die hiervoor als basis dienen, worden ingelezen uit files.
Er zijn drie invoerfiles nodig; twee om de beide blokken apollo's te beschrijven, één voor de algemene gegevens. Deze invoerfiles kunnen in een willekeurige editor gemaakt worden. Wij zullen nu de opbouw van de files bespreken. Allereerst de invoerfile voor het apollo-blok (zie figuur 7.1). In zo'n file staat een aantal apollo's beschreven. Dit kunnen er evenveel zijn als er in werkelijkheid, maar ook minder; in het programma wordt dan met een factor vermenigvuldigd zodanig dat het systeem toch even groot wordt als het werkelijke systeem. Er wordt dus hoe dan ook een systeem gesimuleerd dat even groot is als het systeem bij Bavaria.

In deze voorbeeldfile staan tien apollo's beschreven. Elke beschrijving beslaat vier regels. Op de eerste regel staat welke apollo beschreven wordt. Op de tweede regel staat de soort apollo, 'gist' of 'lager'. Op de derde regel staat in welk proces de apollo begint. Hiervoor zijn de volgende mogelijkheden: vullen, koelingaan, koelinguit, reductie, diepkoelen, verblijf, overpompen, schoonmaken, niet (de apollo doet in het geheel niet mee), uit (de apollo doet pas na een tijd, 'uittijd', mee). Op de laatste regel staan vier getallen. Het eerste getal geeft de inhoud van de apollo, het tweede getal geeft het ingestelde debiet, het derde de 'uittijd' en het laatste getal geeft de grootheid 'alpha'. In dit voorbeeld staan tien apollo's beschreven. In de berekeningen gebruikt het programma het werkelijke aantal apollo's, waarvan er in dit geval tien verschillende 'soorten' zijn. Als de invoerfile maar één apollo beschrijft, rekent het programma met een systeem waarin alle apollo's hetzelfde doen.
Behalve twee invoerfiles van bovengenoemd soort, is er ook nog een algemene invoerfile nodig (zie figuur 7.2).

Deze file bestaat uit vijf blokjes van twee regels. Op de eerste regel daarvan wordt telkens aangegeven wat de getallen op de tweede regel betekenen. In het eerste blokje worden 'Lcond' (de vloeibare ammoniak die op $t = 0$ uit de condensor komt), 'Gretour' (de gasvormige ammoniak die op $t = 0$ uit de condensor komt en wordt teruggevoerd), 'Tbuiten' (de buitentemperatuu op $t = 0$) en de 'Bufinhoud' (de inhoud van de buffer op $t = 0$).
In het tweede blokje worden de afscheiderinhouden (Afschinh[1] en [2]) gegeven op $t = 0$.
In het derde blokje wordt de minimale inhoud van de afscheiders gegeven (mininh[1] en [2]).

Gegevens apollo 1:
gist
koelingaan
180 45.4 10 0.1
Gegevens apollo 2:
gist
diepkoelen
180 45.4 10 0
Gegevens apollo 3:
gist
reductie
180 45.4 10 0.1
Gegevens apollo 4:
gist
koelingaan
180 45.4 10 0.2
Gegevens apollo 5:
gist
uit
180 45.4 10 0
Gegevens apollo 6:
gist
koelingaan
270 52.3 10 0.1
Gegevens apollo 7:
gist
koelingaan
270 52.3 10 0.1
Gegevens apollo 8:
gist
koelingaan
270 52.3 10 0.1
Gegevens apollo 9:
gist
reductie
270 52.3 10 0.1
Gegevens apollo 10:
gist
schoonmaken
270 52.3 10 0.1

Figuur 7.1: Een voorbeeld van een initialisatie-file voor het eerste blok apollo's

Lcond, Gretour, Tbuiten, Bufinhoud
0 0 15 100000
Afschinh[1] en [2]
8000 4000
mininh[1] en [2]
3000 2000
maxinh[1] en [2]
13000 6500
regelinh[1] en [2]
8000 4000

Figuur 7.2: Een voorbeeld van een algemene initialisatie-file

In het vierde blokje staat de maximale inhoud van de afscheiders (maxinh[1] en [2]).
In het vijfde blokje worden de regelinhouden (regelinh[1] en [2]) gegeven.

Wanneer de benodigde invoerfiles aangemaakt zijn, kan het programma gedraaid worden. Om het programma op te starten, moet 'bavkoel' ingetypt worden. Nu verschijnt er een opstartscherm in beeld. Na een druk op een willekeurige toets, komt er een aantal vragen aan de gebruiker.
Allereerst wordt hem gevraagd om de plaats van de bgi-files aan te geven. Deze files heeft het programma nodig om zijn grafische mogelijkheden te benutten. Wanneer men van het schijfje werkt, dient hier 'a:\bgi' ingetypt te worden.
Vervolgens wordt gevraagd naar de naam van de invoerfile met gegevens over het eerste en dan het tweede blok apollo's (bijvoorbeeld 'b1init1.dat' en 'b2init1.dat').
Als vierde dient de naam van de file met algemene gegevens ingetypt te worden (bijvoorbeeld 'init1.dat').
Het programma schrijft zijn uitvoergegevens niet alleen naar het scherm, maar ook naar een file, zodat de gegevens nog op een andere manier bewerkt kunnen worden. Er wordt dan ook gevraagd naar de naam van de file waar deze gegevens naartoe geschreven kunnen worden.
Als laatste moet de simulatieduur (in uren) aangegeven worden.
Als dat gebeurd is verschijnt er een aantal gegevens op het scherm. Dit zijn de gegevens zoals die zijn ingelezen uit de invoerfiles. Om tot de daadwerkelijke simulatie over te gaan, drukt men een willekeurige toets in.
Er verschijnen drie kaders in beeld. In het bovenste kader wordt de inhoud van de eerste afscheider als functie van de tijd geplot; in het middelste kader de inhoud van de tweede afscheider. De grijze lijnen die in deze kaders zijn getekend geven de maximale en minimale niveaus aan; als dit niveau bereikt wordt in het werkelijke systeem, wordt er ingegrepen. In deze simulatie gaan wij gewoon door. De simulatie stopt pas als de afscheiders helemaal leeg zijn.
In het onderste kader worden drie grootheden weergegeven. De hoeveelheid gas die uit beide apolloblokken komt, wordt in oranje geplot; de buitentemperatuur in grijs en de

bufferinhoud in blauw.
Het is mogelijk om tijdens de simulatie de buitentemperatuur te veranderen. De buitentemperatuur wordt hoger als men de 'h' intypt en lager als men de 'l' intypt. Het resultaat van dit ingrijpen is direct te zien in het onderste kader. Als de buitentemperatuur hoger wordt, wordt de condensor slechter. Een lagere buitentemperatuur geeft een betere condensor.
Na beëindiging van de simulatie verwacht het programma een harde return, waarna het programma stopt. Voor een nieuwe simulatie dient wederom 'bavkoel' ingetypt te worden.

A Gegevens

Vergisting (bij pils) levert $6.53 \cdot 10^4$ J/s **(per apollo)**
Er vergist 10 $g/100$ ml glucose in 5 dagen, dus 2 $g/100ml$ glucose per dag. 1 g glucose levert 752.4 J. De inhoud van de apollo's is 5000 hl, die gevuld zijn met 3750 hl wort.

Vergisting (bij zwaar bier) levert $8.16 \cdot 10^4$ J/s **(per apollo)**
Bij zwaar bier vergist er 2.5 $g/100$ ml glucose in plaats van 2 $g/100$ ml

Lagering (bij pils) levert $1.26 \cdot 10^4$ J/s

Lagering (bij zwaar bier) levert $1.74 \cdot 10^4$ J/s

Verdampingswarmte van NH_3, $L = 1.369 \cdot 10^6$ J/kg

Dichtheid van vloeibaar NH_3, $\rho = 640$ kg/m^3

Soortelijke warmte van vloeibaar NH_3,$c_L = 77 \cdot 10^3$ $J/kg \cdot K$

Soortelijke warmte van gasvormig NH_3, $c_G = 2.06 \cdot 10^3$ $J/kg \cdot K$

Soortelijke warmte van wort, $c_{wort} = 1.881 \cdot 10^6 J/K \cdot m^3$
Als het wort binnenkomt duurt het 3 à 4 uur totdat het wort is opgewarmd van 10 $°C$ tot 11 $°C$. Wij gaan ervan uit dat het gist gedurende drie uur actief is en dan $7.05 \cdot 10^8 J$ produceert. Hiermee is de soortelijke warmte van het wort te berekenen.

Als de koeling stopt bij vergisting duurt het 72 $min.$ **voordat de koeling weer aanslaat, hetgeen in overeenstemming is met de ervaring van Bavaria**
De koeling start bij 11.2 $°C$ en stopt bij 10.8 $°C$, dus $\Delta T = 0.4$ K.

Als de koeling stopt bij lagering duurt het ongeveer 4 uur **voordat de koeling weer aanslaat**

Instelgegevens (vast) voor de volume-debieten van de apollo's:

A1-6 425 l/u=1.18 $\cdot 10^{-4} m^3/s$ $\rightarrow D = 7.56 \cdot 10^{-2} g/s$

A7-24 525 l/u=1.46 $\cdot 10^{-4} m^3/s$ $\rightarrow D = 9.35 \cdot 10^{-2} g/s$

A25-33 675 l/u=1.88 $\cdot 10^{-4} m^3/s$ $\rightarrow D = 1.20 \cdot 10^{-1} g/s$

B Programma-listing

```
PROGRAM Bavkoel;

USES graph,crt,jasio;

CONST
  koelguit_eind= 7;    {een tijdseenheid is 10 min.}
  koelluit_eind= 24;   {tijd dat koeling uit staat bij gisten en lageren}
  vul_eind= 30;        {vultijd}
  red_eind=288;        {reductietijd}
  pomp_eind=30;        {overpomptijd}
  schoon_eind= 12;     {schoonmaaktijd}
  hoofd_eind=720;      {totale gisttijd}
  verblijf_eind=576;   {verblijftijd na lageren in lagerapollo's}
  diepg_eind=144;      {diepkoeltijd gistapollo's}
  diepl_eind=288;      {diepkoeltijd lagerapollo's}
  dampwarmte=1.369e6; {NH3:  warmte opgenomen bij verdampen; J/kg}
  gistwarmte=3.918e7; {wort: warmte afgestaan bij gisten;    J/10min}
  lagerwarmte=7.56e6; {wort: warmte afgestaan bij lageren;   J/10min}
  soortwort=1.881e6;  {wort: soortelijke warmte;             J/Km3}
  nomcap=3900;         {nom. capaciteit condensor; kg/10min}
  maxi=100;            {maximaal aantal apollo's per blok}
  Totafsch1=15625;     {totale inhoud van afscheider 1; kg}
  Totafsch2=7969;      {idem voor afscheider 2}
  Totbuffer=8e3;       {idem voor de buffer}
  inh1bav=17888;       {90% van de totale inhoud van blok 1 bij Bavaria,m3}
  inh2bav=9945;        {idem voor blok2}
  rho=640;             {vloeibaar NH3: dichtheid; kg/m3}
  maxgas=6000;         {schaalfactor voor condensorcapaciteit; kg/10min}
  maxtemp=45;          {schaalfactor voor buitentemperatuur; graden C}

TYPE
  procestoestand      = (uit, vullen, koelingaan, koelinguit,
                         reductie, diepkoelen, overpompen,
                         schoonmaken, verblijven, niet);
  gistoflager         = (gist,lager);
  toestand            = RECORD            {toestand van 1 apollo}
                          gistlager     : gistoflager;
                          proces        : procestoestand;
                          koeling       : boolean;
                          teller        : integer;
                          omslagtijd    : integer;
                          inhoud        : real;
                          debiet        : real;
                          koelaan       : boolean;
                          uittijd       : real;
                          temperatuur   : real;
                          alpha         : real;
                        END;
  apollos             = ARRAY [1..2,1..maxi] OF toestand;  {alle apollo's}
  eentwee             = ARRAY [1..2] OF real;
  eentweevijf         = ARRAY [1..5] OF integer;
  punten              = RECORD
                          x: eentweevijf;
                          y: eentweevijf;
                        END;

VAR
  gl_blokje                                  : boolean;
  uitvoer                                    : text;
  blok, blokje, blokjes                      : apollos;
  Lcond, Gretour, Tbuiten, Bufinhoud,
  voldeb, fractie, inh, Gcond                : real;
  blokaant                                   : ARRAY [1..2] OF integer;
  ixpos                                      : ARRAY [1..5] OF real;
  Lin, Luit, Lin_oud, Luit_oud, Guit,
  Guit_oud, Buit_oud, Buit, Regelverschil,
  Afschinh, mininh, maxinh, regelinh,
  convfactor                                 : eentwee;
  maxaant, gd, gm, nr, einde, cst1, cst2,
  n, p, i, j, einduur                        : integer;
  bgi_path, bl1_path, bl2_path,
  str1, init_path, uit_path                  : string;
  lp                                         : punten;
```

```
  Kleur                                    : eentweevijf;


(* procedures en functies *)

PROCEDURE Maximum(bl, b2: integer; VAR blokjes: apollos);
{berekent in welk blok de meeste apollo's staan. Zet in het andere blok
evenveel apollo's neer, waarvan het verschilaantal 'niet' wordt}

VAR
  verschil              : integer;

BEGIN
  IF bl>b2 THEN
  BEGIN
    maxaant:=bl;
    verschil:=bl-b2;
    FOR j:=1 TO verschil DO
    blokjes[2,j+b2].proces:=niet;
  END
  ELSE
  BEGIN
    maxaant:=b2;
    verschil:=b2-bl;
    FOR j:=1 TO verschil DO
    blokjes[1,j+bl].proces:=niet;
  END;
END;


PROCEDURE Kleinnaargroot(VAR cfl, cf2: real);
{berekent de conversiefactoren, nodig om het hele systeem te simuleren}

VAR
  inhoud1, inhoud2       : real;
  k                      : integer;

BEGIN
  inhoud1:=0;
  inhoud2:=0;
  FOR k:=1 TO blokaant[1] DO
  inhoud1:=inhoud1+blok[1,k].inhoud;
  FOR k:=1 TO blokaant[2] DO
  inhoud2:=inhoud2+blok[2,k].inhoud;
  cf1:=inh1bav/inhoud1;
  cf2:=inh2bav/inhoud2;
END;


PROCEDURE Vectortrans(VAR blokje: apollos);
{transformeert apollotoestanden naar koeling aan/uit}

BEGIN
  FOR i:=1 TO 2 DO
  BEGIN
    FOR j:=1 TO maxaant DO
    BEGIN
      CASE blokje[i,j].proces OF
      uit, vullen, koelinguit, reductie,
      overpompen, schoonmaken, verblijven  : blokje[i,j].koeling:=false;
      koelingaan, diepkoelen               : blokje[i,j].koeling:=true;
      END;
    END;
  END;
END;


PROCEDURE Init (VAR blokje: apollos);
{leest invoerfiles en bepaalt totale begintoestand}

VAR
  k                      : integer;
  a, b                   : ARRAY [1..2] OF string;
  c, d, e, f             : ARRAY [1..2] OF real;
  bestand                : ARRAY [1..2] OF text;
  initbestand            : text;

BEGIN
  Clrscr;
  Writeln('geef naam en pad van de file met de gegevens van blok 1...');
  Readln(bl1_path);
  Assign(bestand[1], bl1_path);
  Writeln('geef naam en pad van de file met de gegevens van blok 2...');
  Readln(bl2_path);
  Assign(bestand[2], bl2_path);
  Reset(bestand[1]);
  Reset(bestand[2]);
  blokaant[1]:=0;
  blokaant[2]:=0;
  j:=1;
  FOR k:=1 TO 2 DO
  BEGIN
    REPEAT
      blokaant[k]:=blokaant[k]+1;
      Readln(bestand[k]);
      Readln(bestand[k], a[k]);
      Readln(bestand[k], b[k]);
      Readln(bestand[k], c[k], d[k], e[k], f[k]);
      IF a[k]='gist' THEN blokje[k,j].gistlager:=gist
      ELSE blokje[k,j].gistlager:=lager;
      IF b[k]='niet' THEN blokje[k,j].proces:=niet;
      IF b[k]='uit' THEN blokje[k,j].proces:=uit;
      IF b[k]='vullen' THEN blokje[k,j].proces:=vullen;
      IF b[k]='koelingaan' THEN blokje[k,j].proces:=koelingaan;
      IF b[k]='koelinguit' THEN blokje[k,j].proces:=koelinguit;
      IF b[k]='reductie' THEN blokje[k,j].proces:=reductie;
      IF b[k]='overpompen' THEN blokje[k,j].proces:=overpompen;
      IF b[k]='diepkoelen' THEN blokje[k,j].proces:=diepkoelen;
      IF b[k]='schoonmaken' THEN blokje[k,j].proces:=schoonmaken;
      IF b[k]='verblijven' THEN blokje[k,j].proces:=verblijven;
      blokje[k,j].inhoud:=c[k];
      blokje[k,j].debiet:=d[k];
      blokje[k,j].uittijd:=e[k];
      blokje[k,j].alpha:=f[k];
      j:=j+1;
    UNTIL EOF(bestand[k]);
    Close(bestand[k]);
    j:=1;
  END;
  Maximum(blokaant[1], blokaant[2], blokje);
  Kleinnaargroot(convfactor[1],convfactor[2]);
  FOR i:=1 TO 2 DO
  BEGIN
    Lin[i]:=0; Luit[i]:=0; Lin_oud[i]:=0; Luit_oud[i]:=0;
    Guit[i]:=0; Guit_oud[i]:=0;
    Buit_oud[i]:=0; Buit[i]:=0;
  END;
  Writeln('geef naam en pad van de file met de overige invoergegevens...');
  Readln(init_path);
  Assign(initbestand, init_path);
  Reset(initbestand);
  Readln(initbestand);
  Readln(initbestand, Gretour, Tbuiten, Bufinhoud);
  Readln(initbestand);
  Readln(initbestand, Afschinh[1], Afschinh[2]);
  Readln(initbestand);
  Readln(initbestand, mininh[1], mininh[2]);
  Readln(initbestand);
  Readln(initbestand, maxinh[1], maxinh[2]);
  Readln(initbestand);
  Readln(initbestand, regelinh[1], regelinh[2]);
  Close(initbestand);
  Writeln('geef naam en pad van de file voor de uitvoergegevens...');
  Readln(uit_path);
  Assign(uitvoer, uit_path);
  Rewrite(uitvoer);
```

```
  Writeln('geef de duur van de simulatie in uren');
  Readln(einduur);
  einde:=einduur*6;
  Clrscr;
  Writeln('in blok1 staan ',blokaant[1],' verschillende apollo(s)');
  Writeln('in blok2 staan ',blokaant[2],' verschillende apollo(s)');
  Writeln;
  Writeln('de buitentemperatuur is ',Tbuiten:1:1,' graden');
  Writeln('de bufferinhoud is ',Bufinhoud:4:0,' kg');
  Writeln('de inhoud van afscheider 1 is ',Afschinh[1]:5:0,' kg');
  Writeln('de inhoud van afscheider 2 is ',Afschinh[2]:5:0,' kg');
  Readln;
  InitGraph(gd,gm,bgi_path);
  Kleur[1]:=10;
  Kleur[2]:=10;
  Kleur[3]:=5;
  Kleur[4]:=6;
  Kleur[5]:=8;
  FOR nr:=1 TO 5 DO
  BEGIN
    lp.x[nr]:=3;
    CASE nr OF
    1: lp.y[nr]:=Round(154-152*afschinh[1]/totafsch1);
    2: lp.y[nr]:=Round(313-152*afschinh[2]/totafsch2);
    3: lp.y[nr]:=Round(470-cst2-(150-cst2)*bufinhoud/totbuffer);
    4: lp.y[nr]:=Round(470-cst2-(150-cst2)*Gcond/maxgas);
    5: lp.y[nr]:=Round(470-cst2-(150-cst2)*Tbuiten/maxtemp);
    END;
  END;
  FOR i:=1 TO 2 DO
  BEGIN
    Vectortrans(blok);
    FOR j:=1 TO maxaant DO
    BEGIN
      IF blokje[i,j].koeling THEN
      BEGIN
        Luit_oud[i]:=Luit_oud[i] + blokje[i,j].alpha*blokje[i,j].debiet;     {tot. vl.deb.
 eruit}
        Guit_oud[i]:=Guit_oud[i] + (1-blokje[i,j].alpha)*blok[i,j].debiet;
      END;
    END;
    Buit_oud[i]:=Guit_oud[i];
  END;
END;


PROCEDURE Procesverloop (VAR blokje:apollos);
{geeft overgangen/tijdverlopen apollotoestanden}

BEGIN
  FOR i := 1 TO 2 DO
  BEGIN
    FOR j:=1 TO maxaant DO
    BEGIN
      blokje[i,j].teller := blokje[i,j].teller+1;
      CASE blokje[i,j].proces OF
      niet:
        blokje[i,j].koeling:=false;
      uit:
        IF blokje[i,j].teller = blokje[i,j].uittijd THEN
        BEGIN
          blokje[i,j].proces:=vullen;
          blokje[i,j].teller:=0;
        END;
      vullen:
        IF blokje[i,j].teller=vul_eind THEN
        BEGIN
          blokje[i,j].proces:=koelingaan;
          blokje[i,j].teller:=0;
        END;
      koelingaan:
        BEGIN
          IF blokje[i,j].koelaan=FALSE THEN
          BEGIN
            blokje[i,j].proces := koelinguit;
            blokje[i,j].koelaan := TRUE;
          END;
          IF blokje[i,j].teller = hoofd_eind THEN
          BEGIN
            CASE blokje[i,j].gistlager OF
            gist: blokje[i,j].proces:=reductie;
            lager: blokje[i,j].proces:=diepkoelen;
            END;
            blokje[i,j].teller:=0;
            blokje[i,j].koelaan:=TRUE;
          END;
        END;
      koelinguit:
        BEGIN
          blokje[i,j].omslagtijd:=blokje[i,j].omslagtijd+1;
          CASE blokje[i,j].gistlager OF
          gist:
          BEGIN
            IF blokje[i,j].omslagtijd=koelguit_eind THEN
            BEGIN
              blokje[i,j].proces:=koelingaan;
              blokje[i,j].omslagtijd:=0;
              blokje[i,j].temperatuur:=11.2;
            END;
            IF blokje[i,j].teller=hoofd_eind THEN
            BEGIN
              blokje[i,j].proces:=reductie;
              blokje[i,j].teller:=0;
              blokje[i,j].omslagtijd:=0;
            END;
          END;
          lager:
          BEGIN
            IF blokje[i,j].omslagtijd=koelluit_eind THEN
            BEGIN
              blokje[i,j].proces:=koelingaan;
              blokje[i,j].omslagtijd:=0;
              blokje[i,j].temperatuur:=7.1;
            END;
            IF blokje[i,j].teller=hoofd_eind THEN
            BEGIN
              blokje[i,j].proces:=diepkoelen;
              blokje[i,j].teller:=0;
              blokje[i,j].omslagtijd:=0;
            END;
          END;
          END;
        END;
      reductie:
        IF blokje[i,j].teller=red_eind THEN
        BEGIN
          blokje[i,j].proces:=diepkoelen;
          blokje[i,j].teller:=0;
        END;
      diepkoelen:
        CASE blokje[i,j].gistlager OF
        gist:
        IF blokje[i,j].teller=diepg_eind THEN
        BEGIN
          blokje[i,j].proces:=overpompen;
          blokje[i,j].teller:=0;
        END;
        lager:
        IF blokje[i,j].teller=diepl_eind THEN
        BEGIN
          blokje[i,j].proces:=verblijven;
          blokje[i,j].teller:=0;
        END;
        END;
      verblijven:
        IF blokje[i,j].teller=verblijf_eind THEN
```

```
        BEGIN
          blokje[i,j].proces:=overpompen;
          blokje[i,j].teller:=0;
        END;
      overpompen:
        IF blokje[i,j].teller=pomp_eind THEN
        BEGIN
          blokje[i,j].proces:=schoonmaken;
          blokje[i,j].teller:=0;
        END;
      schoonmaken:
        IF blokje[i,j].teller=schoon_eind THEN
        BEGIN
          blokje[i,j].proces:=vullen;
          blokje[i,j].teller:=0;
        END;
      END;
    END;
  END;
END;


PROCEDURE Alarm;
{stopt het programma als de afscheiders vol of leeg raken}

BEGIN
  IF Afschinh[1]<(0.01*Totafsch1) THEN
  BEGIN
    readln;
    CloseGraph;
    writeln('Afscheider 1 is leeg');
    n:=einde;
  END;
  IF Afschinh[2]<(0.01*Totafsch2) THEN
  BEGIN
    readln;
    CloseGraph;
    writeln('Afscheider 2 is leeg');
    n:=einde;
  END;
  IF Afschinh[1]>(0.99*Totafsch1) THEN
  BEGIN
    readln;
    Closegraph;
    writeln('Afscheider 1 is vol');
    n:=einde;
  END;
  IF Afschinh[2]>(0.99*Totafsch2) THEN
  BEGIN
    readln;
    CloseGraph;
    writeln('Afscheider 2 is vol');
    n:=einde;
  END;
END;


FUNCTION DeltaT (VAR gl_blokje:gistoflager;
         fractie:real; voldeb:real; inh:real): real;
{berekent temp.verandering van wort tijdens gisten en lageren}

BEGIN
  CASE gl_blokje OF
  gist:
  DeltaT:=((1-fractie)*voldeb*dampwarmte-gistwarmte)/
  (inh*soortwort);
  lager:
  DeltaT:=((1-fractie)*voldeb*dampwarmte-lagerwarmte)/
  (inh*soortwort);
  END;
END;
```

```
PROCEDURE Afscheider(VAR debLin: eentwee; VAR inhafsch: eentwee; q:integer;
                     blokje: apollos; oudLuit: eentwee; oudBuit: eentwee);
{berekent grootheden in een afscheider}

BEGIN
  FOR j:=1 TO maxaant DO
  BEGIN
    IF blokje[q,j].koeling THEN
    BEGIN
      debLin[q]:=debLin[q] + blokje[q,j].debiet;
    END;
  END;
  inhafsch[q]:=inhafsch[q]+convfactor[q]*(oudLuit[q]-debLin[q])+oudBuit[q];
  IF (inhafsch[q]<mininh[q]) OR (inhafsch[q]>maxinh[q]) THEN Alarm;
END;


PROCEDURE Apollo(VAR blokje: apollos; VAR deblin, debluit, debguit,
                 deblinoud, debluitoud, debguitoud: eentwee; q: integer);
{berekent grootheden in een apolloblok}

BEGIN
  deblinoud[q]:=deblin[q]; debluitoud[q]:=debluit[q];
  debguitoud[q]:= debguit[q];
  deblin[q]:=0; debluit[q]:=0; debguit[q]:=0;
  FOR j:=1 TO maxaant DO
  BEGIN
    IF blokje[q,j].koeling THEN
    BEGIN
      Luit[q]:=Luit[q] + blokje[q,j].alpha*blokje[q,j].debiet;
      Guit[q]:=Guit[q] + (1-blokje[q,j].alpha)*blok[q,j].debiet;
      IF blokje[q,j].proces=koelingaan THEN
      BEGIN
        blokje[q,j].temperatuur:=blokje[q,j].temperatuur-
        DeltaT(blokje[q,j].gistlager, blokje[q,j].alpha,
        blokje[q,j].debiet, blokje[q,j].inhoud);
        CASE blokje[q,j].gistlager OF
        gist:
        IF blokje[q,j].temperatuur<=10.8 THEN blokje[q,j].koelaan:=false;
        lager:
        IF blokje[q,j].temperatuur<=6.9 THEN blokje[q,j].koelaan:=false;
        END;
      END;
    END;
  END;
END;


PROCEDURE Vergisting(VAR vollin, volguit, volluit, inhoudaf, volluitoud,
                     volbuitoud, vollinoud, volguitoud: eentwee;
                     VAR blokjes: apollos);
{bestuurt afscheider en apolloblok in gistgedeelte}

BEGIN
  Afscheider(vollin, inhoudaf, 1, blokjes, volluitoud, volbuitoud);
  Apollo(blokjes, vollin, volluit, volguit, vollinoud, volluitoud,
         volguitoud, 1);
END;


PROCEDURE Lagering(VAR vollin, volguit, volluit, inhoudaf, volluitoud,
                   volbuitoud, vollinoud, volguitoud: eentwee;
                   VAR blokjes: apollos);
{bestuurt afscheider en apolloblok in lagergedeelte}

BEGIN
  Afscheider(vollin, inhoudaf, 2, blokjes, volluitoud, volbuitoud);
  Apollo(blokjes, volLin, volLuit, volGuit, volLinoud, volLuitoud,
         volGuitoud, 2);
END;


PROCEDURE Condensor(VAR deblcond, debGretour: REAL; VAR debguitoud: eentwee);   {berekent
```

```
grootheden in condensor}
{berekent capaciteit en andere grootheden in de condensor}

TYPE
  weertoestand          = (low, normal, high, veryhigh, extreme);

VAR
  tempcond              : weertoestand;
  Capcond               : real;
  Gasaanvoer            : real;

BEGIN
  IF Tbuiten<5 THEN tempcond:=low;
  IF (Tbuiten>=5) AND (Tbuiten<15) THEN tempcond:=normal;
  IF (Tbuiten>=15) AND (Tbuiten<25) THEN tempcond:=high;
  IF (Tbuiten>=25) AND (Tbuiten<35) THEN tempcond:=veryhigh;
  IF Tbuiten>=35 THEN tempcond:=extreme;
  CASE tempcond OF
  low : Capcond := 1.2*nomcap;
  normal : Capcond := nomcap;
  high : Capcond := 0.9*nomcap;
  veryhigh : Capcond := 0.7*nomcap;
  extreme : Capcond := 0.5*nomcap;
  END;
  Gcond:=convfactor[1]*debGuitoud[1] + convfactor[2]*debGuitoud[2];
  Gasaanvoer:=Gcond + debGretour;
  IF Gasaanvoer >= Capcond THEN
  BEGIN
    debLcond:=Capcond;
    debGretour:=Gasaanvoer - Capcond;
  END;
  IF Gasaanvoer < Capcond THEN
  BEGIN
    debLcond:=Gasaanvoer;
    debGretour:=0;
  END;
END;


PROCEDURE Buffer(VAR Bufinh: real; inhafsch: eentwee; VAR regverschil,
                 debbuit, debbuitoud :eentwee);
{berekent grootheden in buffervat}

BEGIN
  Bufinh:=Bufinh+Lcond;
  FOR i:=1 TO 2 DO
  BEGIN
    debBuitoud[i]:=debBuit[i];
    Regverschil[i]:=Regelinh[i]-inhafsch[i];
    IF Regverschil[i]<0 THEN Regverschil[i]:=0;
  END;
  IF Regverschil[1]+Regverschil[2]<=Bufinh THEN
  BEGIN
    debBuit[1]:=Regverschil[1];
    debBuit[2]:=Regverschil[2];
  END
  ELSE
  BEGIN
    IF Regverschil[1]+Regverschil[2]=0 THEN
    BEGIN
      debBuit[1]:=0;
      debBuit[2]:=0;
    END
    ELSE
    BEGIN
      debBuit[1]:=Regverschil[1]*TotAfsch1*Bufinh/
                  (Regverschil[1]*TotAfsch1+Regverschil[2]*TotAfsch2);
      debBuit[2]:=Regverschil[2]*TotAfsch2*Bufinh/
                  (Regverschil[1]*TotAfsch1+Regverschil[2]*TotAfsch2);
    END;
  END;
  Bufinh:=Bufinh-debBuit[1]-debBuit[2];
END;
```

```
PROCEDURE Condensatie(VAR volLcond, volGretour, bufferinh: real;
                      VAR volguitoud, inhoudaf, regelversch, volbuit,
                      volbuitoud : eentwee);
{bestuurt condensor en vloeistofvat}

BEGIN
  Condensor(volLcond, volGretour, volGuitoud);
  Buffer(Bufferinh, inhoudaf, Regelversch, volbuit, volbuitoud);
END;


PROCEDURE grafiekjes(positie: Integer; ywaarde: real; color: Integer);
{tekent grafieken van relevante grootheden op het scherm}

BEGIN
  SetColor(color);
  MoveTo(lp.x[positie],lp.y[positie]);
  ixpos[positie] := ixpos[positie]+(637-cst1)/einde;
  IF ixpos[positie] >=
  lp.x[positie]+1 THEN
  BEGIN
    lp.x[positie]:=lp.x[positie]+1;
    CASE positie OF
    1:
    BEGIN
      lp.y[positie] := 154-Round(152*ywaarde/Totafsch1);
      LineTo(lp.x[positie],lp.y[positie]);
    END;
    2:
    BEGIN
      lp.y[positie] := 313-Round(152*ywaarde/Totafsch2);
      LineTo(lp.x[positie],lp.y[positie]);
    END;
    3:
    BEGIN
      lp.y[positie] := (470-cst2)-Round((150-cst2)*ywaarde/Totbuffer);
      LineTo(lp.x[positie],lp.y[positie]);
    END;
    4:
    BEGIN
      lp.y[positie] := (470-cst2)-Round((150-cst2)*ywaarde/maxgas);
      LineTo(lp.x[positie],lp.y[positie]);
    END;
    5:
    BEGIN
      lp.y[positie] := (470-cst2)-Round((150-cst2)*ywaarde/maxtemp);
      LineTo(lp.x[positie],lp.y[positie]);
    END;
    END;
  END;
END;


PROCEDURE Kader(kleurtje, cnst1, cnst2: integer);
{tekent het kader voor de grafieken}

BEGIN
  SetColor(kleurtje);
  MoveTo(2,2);
  LineTo(638-cnst1,2);
  LineTo(638-cnst1,154);
  LineTo(2,154);
  LineTo(2,2);
  MoveTo(2,161);
  LineTo(638-cnst1,161);
  LineTo(638-cnst1,313);
  LineTo(2,313);
  LineTo(2,161);
  MoveTo(2,320);
  LineTo(638-cnst1,320);
  LineTo(638-cnst1,470-cst2);
  LineTo(2,470-cnst2);
```

```
  LineTo(2,320);
  SetColor(8);
  SetLineStyle(DottedLn,$aaaa,NormWidth);
  MoveTo(3,Round(154-152*maxinh[1]/Totafsch1));
  LineTo(637-cnst1,Round(154-152*maxinh[1]/Totafsch1));
  MoveTo(3,Round(154-152*mininh[1]/Totafsch1));
  LineTo(637-cnst1,Round(154-152*mininh[1]/Totafsch1));
  MoveTo(3,Round(313-152*maxinh[2]/Totafsch2));
  LineTo(637-cnst1,Round(313-152*maxinh[2]/Totafsch2));
  MoveTo(3,Round(313-152*mininh[2]/Totafsch2));
  LineTo(637-cnst1,Round(313-152*mininh[2]/Totafsch2));
  SetLineStyle(SolidLn,$aaaa,NormWidth);
  SetColor(kleur[1]);
  MoveTo(10,10);
  OutText('inhoud afscheider 1 [m3]');
  SetColor(kleur[2]);
  MoveTo(10,169);
  OutText('inhoud afscheider 2 [m3]');
  SetColor(kleur[3]);
  MoveTo(10,330);
  OutText('inhoud buffer [m3]');
  SetColor(6);
  OutText(' hoeveelheid gas [kg]');
  SetColor(8);
  OutText(' buitentemperatuur');
  SetLineStyle(DottedLn,$aaaa,NormWidth);
  MoveTo(420-cnst1,12);
  LineTo(440-cnst1,12);
  MoveTo(440-cnst1,10);
  OutText(' kritische niveaus');
  SetLineStyle(SolidLn,$aaaa,NormWidth);
  SetColor(Kleur[1]);
  MoveTo(637-cnst1+5,2);
  Str((Totafsch1/rho):1:2,str1);
  OutText(str1);
  SetColor(Kleur[2]);
  MoveTo(637-cnst1+5,161);
  Str((Totafsch2/rho):1:2,str1);
  OutText(str1);
  SetColor(Kleur[3]);
  MoveTo(637-cnst1+5,320);
  Str((Totbuffer/rho):1:2,str1);
  OutText(str1);
  SetColor(Kleur[4]);
  MoveTo(637-cnst1+5,330);
  Str(maxgas,str1);
  OutText(str1);
  SetColor(Kleur[1]);
  MoveTo(637-cnst1+5,154-6);
  OutText('0');
  SetColor(Kleur[2]);
  MoveTo(637-cnst1+5,313-6);
  OutText('0');
  SetColor(Kleur[3]);
  MoveTo(637-cnst1+5,470-6-cnst2);
  OutText('0');
  SetColor(8);
  MoveTo(637-cnst1+5,Round(154-152*maxinh[1]/Totafsch1)-3);
  Str((maxinh[1]/rho):1:2,str1);
  OutText(str1);
  MoveTo(637-cnst1+5,Round(154-152*mininh[1]/Totafsch1)-3);
  Str((mininh[1]/rho):1:2,str1);
  OutText(str1);
  MoveTo(637-cnst1+5,Round(313-152*maxinh[2]/Totafsch2)-3);
  Str((maxinh[2]/rho):1:2,str1);
  OutText(str1);
  MoveTo(637-cnst1+5,Round(313-152*mininh[2]/Totafsch2)-3);
  Str((mininh[2]/rho):1:2,str1);
  OutText(str1);
  SetColor(1);
  MoveTo(2,470-cnst2+5);
  OutText('0');
  MoveTo(637-cnst1-50,470-cnst2+5);
  Str(einduur,str1);
  OutText(str1); OutText(' uur');
END;


PROCEDURE Instant(VAR buittemp:real; cost2:integer);
{maakt interrupt mogelijk om Tbuiten te wijzigen gedurende de simulatie}

VAR
  keuze                 : CHAR;

BEGIN
  keuze:=ReadKey;
  IF keuze='h' THEN buittemp:=buittemp+10;
  IF keuze='l' THEN buittemp:=buittemp-10;
END;

(* einde procedures en functies *)


(* begin hoofdprogramma *)

BEGIN
  Startupscreen('het Koelsysteem van Bavaria',
                'ir C.F.J. den Doelder & drs F.A. Hanneman', 'april 1995');
  {standaardopstartscherm WvdI}
  Writeln('geef naam en pad van de dir waar de bgi-files staan...');
  Readln(bgi_path);       {nodig voor grafische mogelijkheden}
  cst1:=50; cst2:=10;     {parameters voor kadergrootte}
  Init(blok);             {roep init aan}
  Kader(1, cst1, cst2);   {teken het kader}
  FOR n:=1 TO einde DO    {simulatie tot eindtijd}
  BEGIN
    IF (round(n/6)-round((n-1)/6)) =1 THEN
    Writeln(uitvoer, {n, ' ', Bufinhoud:4:6, ' ',} afschinh[1]:4:6{, ' ',
            afschinh[2]:4:6});   {schrijf naar uitvoerfile}

    Procesverloop(blok);        {bereken nieuwe apollotoestanden}
    Vectortrans(blok);          {bereken benodigde koeling}
    Vergisting(Lin, Guit, Luit, Afschinh, Luit_oud, Buit_oud, Lin_oud,
               Guit_oud, blok); {reken het gistblok door}
    Lagering(Lin, Guit, Luit, Afschinh, Luit_oud, Buit_oud, Lin_oud,
             Guit_oud, blok); {reken het lagerblok door}
    Condensatie(Lcond, Gretour, Bufinhoud, Guit_oud, Afschinh, Regelverschil,
                Buit, Buit_oud); {reken de condensor en buffer door}
    Delay(10);  {bouw een (evt.) vertraging in}
    Grafiekjes(1, Afschinh[1], Kleur[1]); {teken de grafieken}
    Grafiekjes(2, Afschinh[2], Kleur[2]);
    Grafiekjes(3, Bufinhoud, Kleur[3]);
    Grafiekjes(4, Gcond, Kleur[4]);
    Grafiekjes(5, Tbuiten, Kleur[5]);
    IF KeyPressed THEN Instant(Tbuiten, cst2); {controleer interrupt}
  END;
  Readln;
  CloseGraph;
  Close(uitvoer);
END.
```